Escuela Politécnica Superior

# Desarrollo de una aplicación móvil para centros de psicología

Máster Universitario en Desarrollo de Software para Dispositivos Móviles

Trabajo Fin de Máster

Autor:
Flaviu Lucian Georgiu

Tutor Universidad Alicante:
Miguel Ángel Lozano Ortega

Tutor empresa Pixel78:
Iván Luciano Nieto Ruiz

Septiembre 2020

Universitat d'Alacant
Universidad de Alicante

# AUGISTER

*Una aplicación de autorregistros online*

**Autor**

Flaviu Lucian Georgiu

**Tutor empresa Pixel78**

Iván Luciano Nieto Ruiz

**Tutor Universidad de Alicante**

Miguel Ángel Lozano Ortega

*Departamento de Ciencia de la Computación e Inteligencia Artificial*

# Resumen

El trabajo desarrollado consta de una aplicación móvil que sirve de extensión a una aplicación web ya existente para centros de psicología. Se han trasladado sus funcionalidades principales como la realización y gestión de autorregistros, gestión de citas y datos de contacto a un entorno móvil para facilitar su utilización para el paciente.

La aplicación es multiplataforma y tiene en cuenta los datos sensibles que se manejan, garantizando la seguridad de la información mediante la utilización de las tecnologías adecuadas y un sistema de bloqueo de aplicación.

# Abstract

This Master's Final Project consists of a mobile application that serves as an extension to an existing web application for psychology offices. Its main functionalities, such as the creation and management of self-registrations, appointment management, and contact details have been transferred to a mobile environment to facilitate their patients' use.

The application is multiplatform and takes into account the use of sensitive data, guaranteeing the security of the information through the use of appropriate technologies and an application blocking system.

# Índice de contenidos

Resumen ..... 3

Abstract ..... 3

Índice de figuras ..... 9

Índice de tablas ..... 11

1. Introducción, justificación y objetivo general ..... 13

2. Estudio de viabilidad ..... 15

2.1 Lean Canvas ..... 15

3. Planificación ..... 19

4. Estado de la cuestión ..... 21

4.1 Pixel78 ..... 21

4.2 Augister ..... 22

4.3 Estudio de la competencia ..... 24

4.3.1 PsicoReg ..... 24

4.3.2 NovoPsych ..... 26

4.3.3 Mentavio ..... 27

4.3.4 iGrade ..... 29

4.3.5 Conclusiones del estudio ..... 30

5. Objetivo ..... 31

6. Metodología ..... 33

7. Análisis y especificación ..... 35

7.1 Requisitos funcionales ..... 36

7.2 Requisitos no funcionales ..... 42

7.3 Otros requisitos ..... 45

7.4 Diagrama de casos de uso ..... 45

8. Diseño ..... 49

8.1 Diseño de la arquitectura conceptual ..... 49

8.2 Diseño de la arquitectura tecnológica ..... 50
8.2.1 Tecnologías del proyecto actual ..... 51
8.2.2 Tecnologías de la aplicación ..... 52
8.3 Diseño de la persistencia ..... 52
8.3.1 Persistencia del servidor ..... 53
8.3.2 Persistencia de la aplicación ..... 53
8.4 Diseño de la seguridad ..... 55
8.5 Diseño de la API ..... 57
8.6 Diseño de interacción e interfaces ..... 62
8.7 Guía de estilos ..... 72
8.8 Diseño de pruebas y validación ..... 75
9. Implementación ..... 77
9.1 Organización del proyecto ..... 77
9.2 Diseño inicial y maquetación ..... 79
9.2.1 Pantalla de login ..... 79
9.2.2 Pantalla con listado de autorregistros ..... 81
9.2.3 Pantalla de autorregistro ..... 82
9.2.4 Otras pantallas ..... 86
9.2.5 Estilos en Ionic ..... 89
9.3 Flujo de la aplicación ..... 91
9.4 Persistencia de Datos ..... 94
9.5 Bloqueo de la Aplicación y Seguridad ..... 96
9.5.1 PIN ..... 97
9.5.2 Identificación biométrica ..... 98
9.5.3 Alias y contraseña ..... 99
9.5.4 Flujo general ..... 101
9.6 Firebase y Notificaciones Push ..... 102
9.7 Integración con la API de Augister ..... 105

9.8 Publicación en los Marketplaces ..... 106
10. Conclusiones ..... 109
10.1 Estado de la aplicación ..... 109
10.2 Metodología ..... 110
10.3 Ionic ..... 110
11. Trabajo futuro ..... 113
Bibliografía ..... 115

# Índice de figuras

Figura 1. PsicoReg ..... 25
Figura 2. NovoPsych ..... 26
Figura 3. Mentavio ..... 28
Figura 4. iGrade ..... 29
Figura 5. Diagrama de caso de uso 1, usuarios ..... 45
Figura 6. Diagrama de caso de uso 2, autenticación ..... 46
Figura 7. Diagrama de caso de uso 3, acciones del paciente ..... 46
Figura 8. Diagrama de caso de uso 4, acciones del psicólogo ..... 47
Figura 9. Diseño de la arquitectura del proyecto ..... 49
Figura 10. Diseño de la pantalla de Login ..... 63
Figura 11. Diseño de la pantalla de Listado de Autorregistros ..... 64
Figura 12. Diseño de la pantalla de Completar Autorregistro ..... 65
Figura 13. Diseño del Menú Lateral ..... 67
Figura 14. Diseño de la pantalla de Cambiar Contraseña ..... 68
Figura 15. Diseño del acceso por Doble Factor de Autenticación ..... 69
Figura 16. Diseño de los modales personalizados ..... 70
Figura 17. Diseño del SplashScreen ..... 71
Figura 18. Diseño del modal informativo, autorregistros por completar ..... 72
Figura 19. Isotipo de Augister ..... 73
Figura 20. Logotipo de Augister ..... 73
Figura 21. Colores de Augister ..... 74
Figura 22. Raleway, fuente de Augister ..... 75
Figura 23. Pantalla de Login ..... 80
Figura 24. Pantalla de Listado de Autorregistros ..... 81
Figura 25. Pantalla de Autorregistro ..... 82
Figura 26. Tipo de Pregunta: entrada de texto breve ..... 83
Figura 27. Tipo de Pregunta: entrada de número ..... 83
Figura 28. Tipo de Pregunta: porcentaje, un valor entre 0 y 100 ..... 84
Figura 29. Tipo de Pregunta: entrada de texto extenso ..... 84
Figura 30. Tipo de Pregunta: seleccionar un valor entre 0 y 10 ..... 84
Figura 31. Tipo de Pregunta: seleccionar un valor entre 0 y 5 ..... 85
Figura 32. Tipo de Pregunta: hora simple ..... 85

Figura 33. Tipo de Pregunta: hora de inicio y fin ..... 85
Figura 34. Tipo de Pregunta: estado de ánimo ..... 85
Figura 35. Tipo de Pregunta: entrada de texto y selección ..... 86
Figura 36. Tipo de Pregunta: sustancia ..... 86
Figura 37. Pantalla de Datos de Contacto ..... 87
Figura 38. Pantalla de Citas ..... 88
Figura 39. Pantalla de Cambiar Contraseña ..... 89
Figura 40. Pantalla de Listado y Autorregistro con Skeleton ..... 92
Figura 41. Menu Lateral ..... 93
Figura 42. Modales ..... 94
Figura 43. Desbloqueo por PIN ..... 97
Figura 44. Desbloqueo por Huella y FaceID ..... 99
Figura 45. Desbloqueo por Alias y Contraseña ..... 100
Figura 46. Elegir Bloqueo de Aplicación ..... 101
Figura 47. Apps de Augister en Firebase ..... 104
Figura 48. Enviar Notificación Push por Postman ..... 105
Figura 49. Ejemplo de Notificación Push ..... 105
Figura 50. Ficha de Augister en Google Play ..... 107
Figura 51. Augister en TestFlight ..... 108

# Índice de tablas

Tabla 1. Lean Canvas ..... 16
Tabla 2. Planificación temporal del TFM ..... 19
Tabla 3. Planificación del desarrollo ..... 20

# 1. Introducción, justificación y objetivo general

En la actualidad, la psicología es una de las ramas más importantes de la ciencia, ya que nos proporciona numerosas herramientas y técnicas para mejorar la calidad de vida de la gente. Se suele minimizar el rol de la misma en nuestra sociedad, limitándolo a trastornos mentales y a problemas derivados de los mismos, pero su conocimiento se puede aplicar a numerosos campos y tener una aportación muy positiva en la sociedad.

La idea de este Trabajo de Final de Máster surge en la empresa Pixel78, situada en Benidorm. Entre los variados proyectos de la empresa, destaca el desarrollo de una aplicación web para centros de psicología que pretende simplificar y ayudar en la gestión de consultas, pacientes, autorregistros y citas. Esta plataforma está desarrollada completamente para un entorno web, pero se llegó a la conclusión de que una aplicación móvil complementaria mejoraría considerablemente la experiencia de usuario y la usabilidad de la plataforma.

Por lo tanto, este objetivo de este trabajo consiste en hacer realidad esa mejora en forma de una aplicación móvil multiplataforma, complementaria de la plataforma web ya existente, que contenga las funcionalidades necesarias para simplificar tareas clave de la misma.

Esta aplicación estaría dirigida principalmente al perfil de paciente, el cual podría completar los autorregistros asignados por los psicólogos de una manera más cómoda, además de poder consultar los ya completados, pedir cita y consultar información. En cuanto al perfil del psicólogo, el objetivo es tener una versión reducida de las funcionalidades de la plataforma web, teniendo una pequeña gestión de los autorregistros y pacientes.

La forma tradicional de trabajar con este tipo de autorregistros es analógica, en papel y, la mayoría de veces, en la propia cita con el psicólogo. Esta nueva metodología de trabajo permite que el psicólogo pueda recibirlos de una manera más sencilla, pueda generar y consultar estadísticas y, por último, preparar las consultas con los pacientes de una forma más óptima, aprovechado mejor el tiempo.

Considero que este proyecto es una manera perfecta para completar mi formación y poner en práctica los conocimientos adquiridos a lo largo del máster, trabajando en un proyecto real con una temática actual y relevante y utilizando las tecnologías necesarias para desarrollar una aplicación multiplataforma útil y con un buen acabado.

Por último, también es una buena oportunidad para conocer más sobre las leyes que se aplican a las aplicaciones que tratan datos sensibles y sobre los procedimientos necesarios para garantizar su seguridad y privacidad.

# 2. Estudio de viabilidad

Este apartado contiene una pequeña parte del estudio de viabilidad de la empresa, donde se ha analizado el proyecto, se han determinado si sus pretensiones u objetivos son viables, pertinentes o necesarios. Para ilustrar parte de este análisis, se utiliza la siguiente herramienta:

## 2.1 Lean Canvas

El Lean Canvas [17] es una herramienta idónea, ya que es un lienzo de modelo de negocio que permite analizar el proyecto desde diferentes perspectivas de manera sencilla y visual. La *Tabla 1. Lean Canvas* muestra el resultado del mismo.

<table>
<tr>
<td rowspan="2">Problemas<br>- Completar y gestionar autorregistros de forma analógica.<br>- Completar autorregistros en diferido.<br>- Alteración de los datos reales.<br>Alternativas<br>- PsicoReg.<br>- NovoPsych.<br>- Mentavio.<br>- iGrade.</td>
<td>Solución<br>- Almacenamiento y gestión de datos en digital.<br>- Completar autorregistros de manera cómoda y casi inmediata a través de la aplicación.<br>- Visualización y análisis de datos previos a la cita, lo que permite optimizar el tiempo.</td>
<td rowspan="2">Propuesta de valor<br>Augister es un servicio web que ofrece la sencilla gestión de pacientes, autorregistros y citas de centros psicológicos o psicólogos independientes. Además, ofrece una aplicación para hacer más cómoda la realización de funcionalidades clave del servicio web, especialmente el completado de autorregistros por parte de los pacientes.</td>
<td>Ventaja diferencial<br>- Visualización de datos de manera gráfica, estadísticas y la generación de informes.<br>- Una aplicación complementaria.<br>- Interfaces de usuario y experiencia de usuario, basadas en la usabilidad.</td>
<td rowspan="2">Segmento de clientes<br>- Centros de psicología.<br>- Psicólogos independientes.</td>
</tr>
<tr>
<td>Métricas clave<br>- Número total de descargas desde los marketplaces.<br>- Número de registros totales.<br>- Número de centros de psicología totales.<br>- Número de psicólogos totales.<br>- Número de pacientes totales.</td>
<td>Canales<br>- Web corporativa del servicio.<br>- Blogs y foros de psicólogos.<br>- Directorios online de centros de psicología.<br>- RRSS: Facebook e Instagram.<br>- Boca a boca.</td>
</tr>
<tr>
<td colspan="3">Estructura de costos<br>- Desarrollo de la aplicación móvil y del servicio web.</td>
<td colspan="2">Flujos de ingresos<br>- Pago por funcionalidades extra en el perfil del psicólogo.</td>
</tr>
</table>

| | |
|---|---|
| - Hosting.<br>- Dominio.<br>- Publicidad y márquetin.<br>- Publicación en tiendas (App Store y Play Store) y licencias.<br>- Recursos humanos.<br>- Textos legales. | - Distintos planes con un número máximo de pacientes registrados. |

*Tabla 1. Lean Canvas*
*(Fuente propia)*

Para comprender mejor los apartados definidos en la *Tabla 1. Lean Canvas,* se dispone a detallar cada uno de ellos:

### Segmentos de clientes

El nicho de mercado son los centros de psicología o psicólogos independientes que requieran de una sencilla organización de pacientes, citas y autorregistros online.

Aunque la aplicación vaya a ser utilizada también por los pacientes, éstos no son el cliente prototipo. Esto se debe a que el contenido principal es proporcionado y gestionado por cada centro de psicología, lo que provocaría que no tuviera sentido su uso sin el involucramiento de este perfil.

### Problemas

Los clientes que se han mencionado en el apartado anterior, por lo general, tienen los siguientes problemas:

- Completar y gestionar los autorregistros de forma analógica
- Completar autorregistros en diferido.
- Alteración de los datos reales.

### Propuesta de valor

Lo que se propone es Augister, un servicio web que ofrece la sencilla gestión de pacientes, autorregistros y citas de centros psicológicos o psicólogos independientes. Además, ofrece una

aplicación para hacer más cómoda la realización de funcionalidades clave del servicio web, especialmente el completado de autorregistros por parte de los pacientes.

### Solución

Para solucionar los problemas identificados en el apartado de “Problemas”, se propone como soluciones:

- Almacenamiento y gestión de datos digitalmente.
- Completar autorregistros de manera cómoda y casi inmediata a través de la aplicación.
- Visualización y análisis de datos previos a la cita, lo que permite optimizar el tiempo.

### Canales

A la hora de llegar a los potenciales clientes, es necesario determinar los canales idóneos. Teniendo en cuenta las características del proyecto, los principales canales serían las redes sociales (Facebook e Instagram), la propia web corporativa del servicio, blogs y foros de psicólogos, directorios online de centros de psicología y, por último, el boca a boca.

### Flujos de ingresos

Cabe destacar que, en un principio, el servicio sería gratuito con un número limitado de psicólogos y pacientes. Los principales flujos de ingreso serían el pago por funcionalidades extra en el perfil del psicólogo y distintos planes con un número máximo de pacientes registrados.

### Estructura de costos

Para el desarrollo del proyecto se tienen en cuenta una serie de costos. Lo más costoso sería el propio desarrollo de la aplicación móvil y del servicio web, la publicidad y el márquetin, además de todos los recursos humanos involucrados. También es necesario considerar el costo del hosting, el dominio, la publicación de la aplicación en tiendas (App Store y Play Store) y las

distintas licencias necesarias. Por último, para la alineación con la legalidad se tienen en cuenta el costo de consultas a profesionales y la creación de los textos legales.

### Métricas clave

Con el objetivo de verificar y probar el éxito de la aplicación, se disponen las siguientes métricas a tener en cuenta:

- Número total de descargas desde los marketplaces.
- Número de registros totales.
- Número de centros de psicología totales.
- Número de psicólogos totales.
- Número de pacientes totales.

### Ventaja diferencial

Por último, la ventaja diferencial indica el factor que hace al servicio especial y diferente. Para el proyecto en cuestión, son la visualización de datos de manera gráfica, estadísticas y la generación de informes, la aplicación complementaria y el diseño y la usabilidad del servicio.

# 3. Planificación

Para llevar a cabo el proyecto con éxito, es importante dedicar tiempo a la planificación del desarrollo del mismo. En la *Tabla 2. Planificación temporal del TF* se puede ver la planificación que tiene como objetivo asegurar la realización de la defensa del TFM para mediados de septiembre de 2020 (C4). Cada apartado tiene un tiempo y fecha límite aproximada para su desarrollo.

| Contenidos | Tiempo total | Fecha límite fin |
|---|---|---|
| Análisis y especificación<br>Objetivos | 2 semanas | 16 febrero |
| Motivación, justificación, objetivo general, Introducción<br>Presupuesto, estimaciones, planificación<br>Metodología<br>Estado del arte | 3 semanas | 15 marzo |
| Diseño e Implementación | 4 semanas | 17 mayo |
| Pruebas y validación<br>Resultados<br>Conclusiones y trabajo futuro<br>Referencias, bibliografía y apéndices<br>Agradecimientos, citas, índices | 2 semanas | 31 julio |

*Tabla 2. Planificación temporal del TFM*
*(Fuente propia)*

Cabe mencionar que, al ser un proyecto desarrollado para una empresa, la implementación es prioritaria, se comienza casi desde el principio con la misma y se mantiene en paralelo durante el desarrollo de la memoria.

Además, se puede desglosar la implementación en un conjunto de fases, compuestas por una secuencia de diseño, implementación y, finalmente, validación por parte de la empresa. La Tabla 3. Planificación del desarrollo contiene los pasos a seguir y las fechas marcadas por la empresa.

| Contenidos | Tiempo total | Fecha límite fin |
|---|---|---|
| Maquetación y flujo | 3 semanas | 16 febrero |
| Funcionalidades básicas (autorregistros) | 3 semanas | 15 marzo |
| Funcionalidades avanzadas (notificaciones push, seguridad, etc...) | 3 semanas | 19 abril |
| Integración con la API final y ultimar detalles | 4 semanas | 17 mayo |
| Revisión y pruebas | 2 semanas | 31 mayo |
| Pruebas con usuarios reales | 1 mes | 30 junio |
| Publicación en los marketplaces | 1 semana | 6 agosto |

*Tabla 3. Planificación del desarrollo*
*(Fuente propia)*

Cabe destacar que los plazos son bastante amplios porque se tiene en cuenta la redacción de la memoria y la carga de trabajo del alumno en el máster. Las fechas también tienen en cuenta el trabajo de la empresa de mejora de la plataforma actual de Augister, así como una migración de la API vigente.

Debido a la crisis sanitaria actual, hay ciertos huecos entre las semanas planificadas y ha sido necesario volver a planificar algunas de las tareas.

La experiencia recogida al finalizar el trabajo resultará muy valiosa para la planificación de futuros proyectos.

# 4. Estado de la cuestión

A lo largo de este capítulo se procederá a introducir a la empresa responsable de la idea del proyecto, el servicio web ya existente y, finalmente, se hará un pequeño análisis de las aplicaciones y plataformas que ofrecen un servicio similar.

## 4.1 Pixel78

La empresa con la iniciativa para este trabajo es Pixel78 [19], un estudio creativo situado en Benidorm, Alicante. Esta empresa nace a manos de un ingeniero informático, Iván Nieto, un antiguo alumno de la Universidad de Alicante, cuyas inquietudes e intereses van más allá de la ingeniería, teniendo un perfil más creativo y social.

Pixel78 ofrece servicios de programación a medida, desarrollo de aplicaciones para dispositivos móviles, diseño y programación web, posicionamiento en buscadores (SEO y SEM), creación y mantenimientos en redes sociales, consultoría de branding y todo lo relacionado con la publicidad y el diseño gráfico.

El mayor propósito de la empresa es la satisfacción total del cliente, cuidando todo trabajo hasta el último detalle y poniendo a su disposición todo el conocimiento, herramientas y saber a través de un diálogo sincero.

Pixel78 trabaja con una gran variedad de clientes, desde pequeñas empresas locales como grandes entidades reconocidas a nivel nacional. Se pueden nombrar al Ayuntamiento de Benidorm, el Diario Información, la Prensa Ibérica y la cadena de hoteles ServiGroup, entre otros.

Además, la empresa se implica en varios tipos de trabajos, desde proyectos ya existentes a proyectos construidos a medida para cliente desde cero. Asimismo, también se mencionan sus proyectos propios, como puede ser Augister.

Por último, destacar el gran rango de tecnologías que se utilizan para los distintos proyectos de programación, que abarcan desde Node, Laravel, PHP en el backend, Ionic, Angular, HTML 5 y CSS 3 en frontend y el uso de CMS como Wordpress, Prestashop y Magento.

Todo esto es posible gracias a un buen equipo dividido en varios departamentos, como pueden ser la gestión de proyectos, programación y diseño gráfico.

## 4.2 Augister

Augister surge como un proyecto independiente, a raíz de la idea de compromiso social de la empresa y de un socio de la misma, psicólogo de profesión.

Esta aplicación web se puede definir como una herramienta para psicólogos con la cual se puede gestionar de una forma totalmente diferente un centro de psicología, incorporando las nuevas tecnologías. Esto permite, de una forma sencilla, rápida y profesional, crear autorregistros online para cualquier trastorno, situación o actividad, suponiendo un ahorro en papel y tiempo, además de un almacenamiento de datos en formato digital.

Los autorregistros son una herramienta muy importante para la obtención de un buen diagnóstico o tratamiento. Para ello, Augister facilita a los pacientes el acceso inmediato a los autorregistros en los momentos clave. Podrán acceder desde cualquier lugar, siempre que tengan acceso a Internet, y rellenar los autorregistros que tengan asignados. Precisamente esta inmediatez en la introducción de los datos permite ofrecer una información reciente y veraz de todo aquello que le está sucediendo al paciente.

Por lo tanto, una de las mayores ventajas en la utilización de este software es una mejor organización del tiempo del psicólogo, gracias a la información veraz en tiempo real del seguimiento de cada paciente. Estos datos se pueden consultar fácilmente desde el perfil del paciente, pudiendo así enfocar de una manera más eficiente la próxima cita y preparar mejor las siguientes actividades.

Augister permite administrar los autorregistros online de una manera sencilla, creando plantillas con preguntas propias o utilizando plantillas predefinidas de autorregistros. Cabe destacar que estas plantillas predefinidas han sido preparadas y respaldadas por los mejores profesionales y académicos de la actualidad, pero permiten la completa adaptación a las necesidades de cada psicólogo.

Otro punto fuerte de la aplicación es la generación de gráficas a partir de los autorregistros completados y la posibilidad de consultar, con un simple vistazo, la evolución del paciente en cuestión. Esto es posible gracias a un algoritmo propio, desarrollado con el objetivo traducir los textos que introduce el paciente relacionados con sentimientos, sensaciones, estados de ánimo, etc, en gráficas. De esta manera, se le permite al psicólogo comparar los ciclos por periodos de tiempo, siempre y cuando el paciente tenga datos suficientes.

El servicio tiene dos perfiles diferenciados. Por una parte, se tiene al psicólogo, con un perfil de administración con privilegios. Las actividades que puede llevar a cabo son:

- Crear y gestionar pacientes.
- Crear, gestionar autorregistros o utilizar las plantillas predefinidas de la plataforma.
- Asignar autorregistros.
- Visualizar estadísticas y generar informes con los datos de las respuestas a los autorregistros.
- Aceptar y gestionar las citas con los pacientes.

El segundo perfil sería el del paciente, un usuario que únicamente podría:

- Responder a los autorregistros asignados a él por su psicólogo.
- Consultar datos de contacto del centro de psicología o de su psicólogo.
- Pedir cita (tiene que ser confirmada por el psicólogo).
- Cambiar su contraseña.

Esta herramienta está disponible en español, ya que es el público mayoritario a la que va dirigida, pero también está traducida al inglés, lo que permite llegar a un número mayor de clientes.

Desde un punto de vista técnico, el servicio web se trata de un SaaS (Software as a Service), desarrollado principalmente con CodeIgniter (PHP), Bootstrap y JQuery. Además, la plataforma consta de un servidor Apache para hacerse cargo de las peticiones HTTP. También cabe mencionar que se tiene un sistema de notificaciones push, de manera que los pacientes sean notificados cuando se le ha sido asignado un autorregistro nuevo.

Por último, destacar que el servicio es totalmente gratuito para un uso básico y un número limitado de pacientes y psicólogos. Aún así, los principales flujos de ingreso son el pago por funcionalidades extra en el perfil del psicólogo y los distintos planes de precios con un número máximo de pacientes registrados.

## 4.3 Estudio de la competencia

Se procede a analizar las aplicaciones y los servicios que ofrecen un servicio similar.

### 4.3.1 PsicoReg

PsicoReg [22] es una plataforma para psicólogos para la gestión de pacientes, bastante similar a Augister, siendo el principal competidor. Dispone de una aplicación web para la gestión general de los psicólogos y pacientes y una aplicación móvil destinada exclusivamente a pacientes para completar autorregistros.

Además de tener las funcionalidades básicas de gestión, tiene un sistema de videoconferencia para la comunicación online entre el psicólogo y el paciente. En el caso de Augister, tanto esta funcionalidad como la comunicación por mensajería con el psicólogo fueron descartadas, ya que se prioriza las consultas presenciales y las ventajas que esto conlleva.

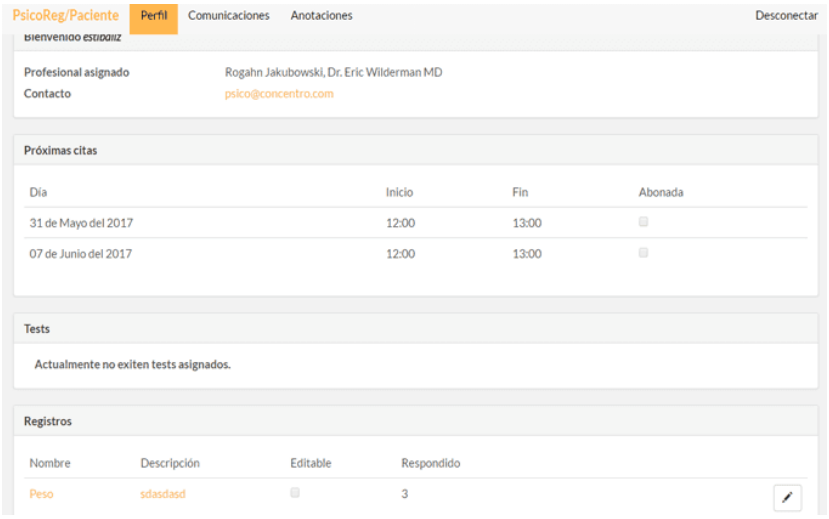


*Figura 1. PsicoReg (psicoreg.com)*

Aunque, después de probarla, cumple con las funcionalidades básicas como es de esperar, hay ciertas desventajas al utilizar este servicio. En primer lugar, el servicio no es multi-idioma, aunque esto no es un gran problema si está dirigido únicamente a personas hispanohablantes.

En cuanto a la generación de estadísticas, éstas se limitan a gráficos basados en "escalas", que son formularios (autorregistros) predefinidos por la plataforma. Al utilizar esta plataforma, las "escalas" tienen que ser enviadas explícitamente por el psicólogo al paciente para poder generar dichas estadísticas. En cambio, Augister genera estadísticas e informes de todos los autorregistros completados por el paciente, ya sea autorregistros basados en "escalas" predefinidas o autorregistros creados o customizados por el psicólogo.

Otro punto en contra es el diseño y la facilidad de uso del mismo. Nos encontramos con un diseño anticuado y poco atractivo. Por otra parte, el servicio consta de una interfaz algo confusa y poco intuitiva. Augister pretende ofrecer un diseño agradable y vigente, además de unas interfaces pensadas para su facilidad de uso.

Cabe mencionar que, la aplicación móvil que complementa al servicio, va dirigida exclusivamente al perfil de paciente con el objetivo de completar autorregistros. Las valoraciones de la misma en Google Play rondan las tres estrellas y media, siendo la queja

principal este aspecto, que la aplicación esté destinada únicamente a este perfil. En cambio, la aplicación móvil de Augister tendrá en cuenta también el perfil de psicólogo, permitiendo una gestión mínima de pacientes, citas y autorregistros.

Por último, mencionar la falta de información sobre el servicio en su propia página web, sobre todo sobre los planes de precios. Lo único que podemos encontrar es una prueba gratuita con un máximo de un usuario psicólogo y dos pacientes.

### 4.3.2 NovoPsych

NovoPsych [26] se trata de otro competidor importante. Este servicio está más enfocado a la realización de autorregistros, teniendo la gestión de pacientes en un segundo plano.

Teniendo este enfoque con los autorregistros, esta plataforma destaca por su robusta y diversa generación de estadísticas y gráficos a partir de las respuestas de los pacientes. Tomando esto en consideración, persigue el mismo objetivo final que Augister, optimizar las sesiones con los pacientes y permitir a los psicólogos enfocar mejor sus tratamientos.

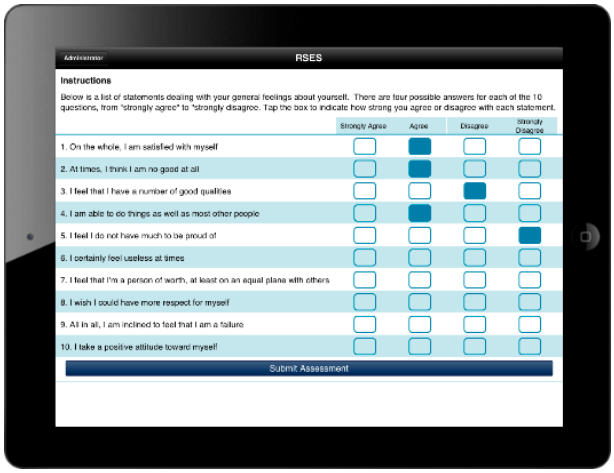


*Figura 2. NovoPsych (novospych.com.au)*

Cabe destacar que este servicio no se encuentra disponible en español, teniendo el inglés como único idioma. Aun así, las principales diferencias con Augister son las que se proceden a definir a continuación.

Como ya se ha comentado anteriormente, la gestión de pacientes y autorregistros pasan a un segundo plano, no siendo tan importantes, lo cual hace que estas funcionalidades estén algo descuidadas.

En cuanto a la disponibilidad del servicio, este consta de una aplicación web y una aplicación móvil exclusiva para iPad. Esto implica que los pacientes con dispositivos iOS y Android deben acceder a sus autorregistros desde el navegador web, siendo esto un inconveniente.

Es importante mencionar la falta de una gestión de citas, lo cual se hará de forma analógica fuera del servicio. En cuanto a usabilidad y diseño, está al nivel del servicio anterior, PsicoReg. Sigue teniendo un diseño poco vigente y una usabilidad en la aplicación web algo descuidada.

Por último, en cuanto a precio, sigue la misma estrategia que Augister, ofreciendo un plan gratuito con un número máximo de psicólogos y pacientes, además de varios planes premium con límites variables de usuarios.

A diferencia del servicio anterior, este está mucho mejor documentado en su página web, lo que le da una mejor imagen e inspira más confianza.

### 4.3.3 Mentavio

Mentavio [21] consta de un servicio para comunicar psicólogos y clientes. Esta plataforma no va dirigido a centros de psicología para la gestión básica de su centro, sino que funciona como un directorio de psicólogos, donde éstos pueden trabajar de forma remota. De esta manera, los pacientes acceden a la plataforma, seleccionan un psicólogo de su agrado, hacen una reserva y

abren un canal de comunicación con el profesional seleccionado, pagando el servicio del mismo por horas como si de una consulta presencial se tratara.

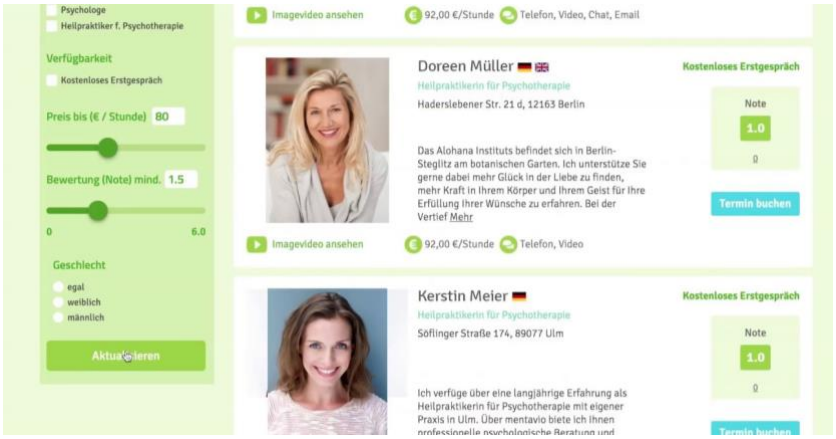


*Figura 3. Mentavio*
*(mentavio.es)*

Además de los diversos canales de comunicación con los pacientes de forma remota, se ofrece la posibilidad de asignar autorregistros sencillos, aunque bastante más limitados que los ofrecidos por los servicios anteriores.

Mentavio se enfoca principalmente desde su aplicación web, pero tiene también una aplicación móvil multiplataforma con las funcionalidades básicas de calendario, acceso al directorio de psicólogos y la comunicación con los mismos a través de diversos canales.

A pesar de tener funcionalidades básicas similares a los servicios ya mencionados, es bastante diferente a Augister, ya que los públicos a los que van dirigidos son diferentes, además de tener un objetivo final totalmente distinto. Mentavio pone énfasis en la comunicación remota, mientras Augister le da más prioridad a los autorregistros y a la mejora de las consultas presenciales.

### 4.3.4 iGrade

iGrade [9] es una aplicación móvil para iPadOS e iOS que ofrece una gestión de los pacientes de un centro de psicología. Aunque no tenga una aplicación web o un sistema de autorregistros, se tiene en cuenta debido a su posición en la AppStore.


*Figura 4. iGrade*
*(Zysco)*

Es una aplicación con un diseño poco cuidado, con una usabilidad que deja mucho que desear, disponible solamente en inglés y a un precio muy elevado. Todo esto contribuye a que tenga una media de dos estrellas de cinco.

A pesar de todo esto, es una aplicación a tener en consideración, ya que la aplicación móvil de Augister tendría una gestión mínima de pacientes, que permitiría al psicólogo crear y consultar los datos de los mismos. Además, Augister solucionaría todos los problemas presentes en esta aplicación, siendo una opción superior.

### 4.3.5 Conclusiones del estudio

Para concluir este pequeño estudio de la competencia, se puede afirmar que ninguno de los servicios vistos tiene el mismo set de funcionalidades que Augister, siendo este último el más completo.

Un aspecto en el que se desmarca Augister es la gestión de autorregistros, ya que es el único servicio de los vistos que permite la creación de autorregistros personalizados por el psicólogo.

Por último, Augister es el servicio con el diseño más moderno y actual, ofreciendo unas interfaces simples y directas y una experiencia de uso más agradable.

# 5. Objetivo

El principal objetivo de este trabajo de fin de máster es desarrollar una aplicación móvil que sirva de extensión a una aplicación web ya existente para centros de psicología, Augister. Los objetivos fundamentales son:

- Desarrollar el perfil para pacientes de la aplicación, permitiéndole a éste el acceso a sus autorregistros y el completado de los mismos de una forma rápida y sencilla.
- Permitir la creación de citas previas con el psicólogo a través de la aplicación.
- Permitir el acceso a datos de contacto del centro al que pertenece el paciente.
- Usar los protocolos y primitivas criptográficas necesarias y adecuadas para garantizar la privacidad, autenticidad, integridad y disponibilidad de la información sensible, dentro de la aplicación.
- Desarrollar una interfaz sencilla e intuitiva para garantizar una experiencia de usuario correcta.

Cabe destacar los siguientes objetivos específicos y secundarios:

- Desarrollar el perfil de psicólogo que le permita tener una gestión mínima de los pacientes, así como de las citas creadas desde la aplicación.
- Desarrollar una gestión mínima de los autorregistros realizados por los pacientes.
- Aprender a utilizar de manera correcta y óptima las herramientas que se usarán para el desarrollo de la aplicación móvil multiplataforma.
- Aprender a adaptar las interfaces de manera adecuada a dispositivos móviles, así como utilizar los controles adecuados para ofrecer la mejor experiencia de usuario.
- Determinar las tecnologías adecuadas para el funcionamiento óptimo y eficiente de la aplicación.
- Analizar y determinar los protocolos y primitivas más adecuadas para garantizar la seguridad de los archivos.
- Desarrollar un sistema de autenticación de doble factor para aumentar la seguridad del acceso a la plataforma.

Como objetivo personal, se desea desarrollar una aplicación útil que englobe los conocimientos vistos a lo largo del máster y pueda ser de utilidad a la sociedad.

# 6. Metodología

Para el desarrollo de este trabajo no se utiliza un modelo en específico, sino una combinación de varios, tomando una parte de cada uno de ellos. Esto se debe a la naturaleza del proyecto, ya que es un desarrollo para una empresa, en el cual la implementación y obtención de resultados es una parte que cobra mucho peso.

Se pensó en un principio en utilizar SCRUM debido a que permite ver resultados rápidamente, aunque éstos sean graduales y no siempre significativos, pero se descartó porque no encajaba en la forma de trabajar para este proyecto. No es posible hacer un seguimiento diario y tampoco tener un producto funcional al finalizar cada una de las iteraciones o sprints.

La combinación de metodologías utilizadas toma, en primer lugar, parte de la secuencia del modelo cascada, siendo esta formada por requisitos, diseño, implementación y verificación. El proyecto comienza con un proceso bastante exhaustivo de análisis y especificación, donde se identifica el problema, los objetivos y las características del mismo. El hecho de partir de un servicio web maduro permite tener los requisitos más claros desde el principio

Se diseña una solución y se pasa al extenso proceso de implementación, donde se pretende mejorar la experiencia del usuario de la web, trasladando las funcionalidades más importantes a una versión móvil. Cuando el proyecto esté listo, se espera una verificación final de los resultados.

Adentrándonos en la fase de implementación, ésta se ha dividido en fases, tomando elementos del modelo OpenUp [18], que permite más flexibilidad y convierte esta parte de la secuencia del modelo cascada en una etapa ágil. OpenUp permite tener un enfoque más pragmático, manteniendo un desarrollo incremental y una colaboración óptima con la empresa, equilibrando las prioridades para maximizar los beneficios obtenidos. Además, se tiene una retroalimentación temprana y continua por parte de la empresa.

Cabe destacar que no se puede seguir estas metodologías a rajatabla, ya que no se dispone de las condiciones necesarias para cumplir con todas las reglas, por lo que se utilizará una variante adaptada.

En cada fase o iteración, se definen las tareas necesarias, se ordenan según su prioridad y se retoman aquellas que no han podido ser realizadas en alguna fase anterior.

Para la organización con la empresa, se ha hecho uso de la herramienta Trello [25]. Destaca por su sencillez a la hora de definir, organizar y gestionar tableros a gusto del usuario.

# 7. Análisis y especificación

Este apartado trata de definir las especificaciones y requerimientos de la aplicación para concretar la solución planteada para este proyecto. Es esencial analizar y diseñar las especificaciones de la aplicación para delimitar y dejar claras sus funcionalidades.

Para ello, se utilizará un modelo simplificado del sistema IEEE830. Este estándar nos proporciona un modelo sencillo que nos permite recoger los requisitos del sistema, desde los relacionados con los usuarios hasta las funcionalidades más específicas, abarcando todas las áreas del proyecto.

Siguiendo el modelo, se han separado los requisitos en 3 tipos:

- **Funcionales (RF)**: generalmente definen funcionalidades o servicios que presta el sistema.
- **No funcionales (RNF)**: generalmente relacionadas con las propiedades del sistema.
- **Otros requisitos** a tener en cuenta **(OR)**.

Cada requisito está representado en una tabla. Cada una contiene los siguientes atributos:

- Un **identificador**, un código alfanumérico que sigue la estructura “tipo-número”.
- **Nombre**.
- **Usuario**.
- **Tipo**: básico (necesario para que el sistema funcione correctamente) o avanzado (funcionalidad adicional).
- **Descripción**.

A lo largo de los requisitos funcionales, los usuarios que aparecen son:

- **Usuario no identificado**.
- **Paciente**, un usuario normal, con un perfil propio y funcionalidades limitadas.

- **Profesional**, generalmente un psicólogo. Este usuario tiene mayores privilegios y acceso a más funcionalidades del sistema.
- **Sistema**, utilizado en funciones realizadas por la misma aplicación.

Por último, comentar que, en cuanto a los requisitos no funcionales y otros requisitos, los atributos de usuario y tipo desaparecen, ya que el usuario de dichos requisitos es el propio sistema y el tipo es, por lo general, básico para el funcionamiento óptimo del sistema.

## 7.1 Requisitos funcionales

| | |
|---|---|
| Identificador | RF-1 |
| Nombre | Iniciar sesión |
| Usuario | Usuario no identificado |
| Tipo | Básico |
| Descripción | El usuario puede iniciar sesión en el sistema utilizando su alias (Fir) () y su contraseña. |

| | |
|---|---|
| Identificador | RF-2 |
| Nombre | Iniciar sesión con doble factor |
| Usuario | Usuario no identificado |
| Tipo | Básico |

| | |
|---|---|
| Descripción | Habilitar el doble factor de autenticación mediante la generación de una clave adicional temporal TOTP propia para cada usuario recibida por correo electrónico. Este requisito es obligatorio para el usuario profesional. |

| | |
|---|---|
| Identificador | RF-3 |
| Nombre | Listar autorregistros por completar |
| Usuario | Paciente |
| Tipo | Básico |
| Descripción | Permitir al paciente consultar los autorregistros que debe completar |

| | |
|---|---|
| Identificador | RF-4 |
| Nombre | Listar autorregistros completados |
| Usuario | Paciente |
| Tipo | Básico |
| Descripción | Permitir al paciente consultar el historial de autorregistros ya completados |

| | |
|---|---|
| Identificador | RF-5 |

| | |
|---|---|
| Nombre | Completar Autorregistros |
| Usuario | Paciente |
| Tipo | Básico |
| Descripción | Permitir al paciente completar los autorregistros creados y asignados por el profesional |

| | |
|---|---|
| Identificador | RF-6 |
| Nombre | Consultar datos de contacto |
| Usuario | Paciente |
| Tipo | Básico |
| Descripción | Permitir al paciente ver los datos de contacto del profesional y su consulta. |

| | |
|---|---|
| Identificador | RF-7 |
| Nombre | Pedir cita |
| Usuario | Paciente |
| Tipo | Básico |
| Descripción | Permitir al paciente pedir cita para acudir a la consulta del profesional |

| | |
|---|---|
| Identificador | RF-8 |
| Nombre | Enviar notificaciones Push |
| Usuario | Sistema |
| Tipo | Básico |
| Descripción | Enviar notificaciones push a los pacientes a modo de recordatorio para completar los autorregistros faltantes. También se recuerda al usuario que tiene una cita programada o se avisa si tiene nuevos autorregistros disponibles. |

| | |
|---|---|
| Identificador | RF-9 |
| Nombre | Consultar pacientes |
| Usuario | Profesional |
| Tipo | Avanzado |
| Descripción | Permitir al profesional consultar la lista de sus pacientes |

| | |
|---|---|
| Identificador | RF-10 |
| Nombre | Crear pacientes |
| Usuario | Profesional |
| Tipo | Avanzado |

| | |
|---|---|
| Descripción | Permitir al profesional crear una cuenta de paciente |

| | |
|---|---|
| Identificador | RF-11 |
| Nombre | Listar citas |
| Usuario | Profesional |
| Tipo | Avanzado |
| Descripción | Permitir al profesional consultar sus próximas citas |

| | |
|---|---|
| Identificador | RF-12 |
| Nombre | Confirmar cita |
| Usuario | Profesional |
| Tipo | Avanzado |
| Descripción | Permitir al profesional confirmar una cita pedida por el paciente |

| | |
|---|---|
| Identificador | RF-13 |
| Nombre | Listar batería de autorregistros |
| Usuario | Profesional |

| | |
|---|---|
| Tipo | Avanzado |
| Descripción | Permitir al profesional consultar los autorregistros que puede asignar |

| | |
|---|---|
| Identificador | RF-14 |
| Nombre | Ver detalle de autorregistro |
| Usuario | Profesional |
| Tipo | Avanzado |
| Descripción | Permitir al profesional ver los detalles y las preguntas definidas para el autorregistro en cuestión. |

| | |
|---|---|
| Identificador | RF-15 |
| Nombre | Asignar Autorregistro |
| Usuario | Profesional |
| Tipo | Avanzado |
| Descripción | Permitir al profesional asignar un autorregistro (ya creado en la web) a sus pacientes |

| | |
|---|---|
| Identificador | RF-16 |
| Nombre | Listar autorregistros asignados |

| | |
|---|---|
| Usuario | Profesional |
| Tipo | Avanzado |
| Descripción | Permitir al profesional consultar los autorregistros ya asignados a sus pacientes |

| | |
|---|---|
| Identificador | RF-17 |
| Nombre | Buscar autorregistros |
| Usuario | Paciente |
| Tipo | Avanzado |
| Descripción | Permitir al profesional buscar autorregistros entre los disponibles y asignados. |

## 7.2 Requisitos no funcionales

| | |
|---|---|
| Identificador | RNF-1 |
| Nombre | Rendimiento |
| Descripción | El tiempo de respuesta del sistema debe ser inferior a los 3 segundos. |

| | |
|---|---|
| Identificador | RNF-2 |

| | |
|---|---|
| Nombre | Encriptación de las interacciones |
| Descripción | Toda interacción cliente – servidor debe ser realizada a través de un canal seguro, usando el protocolo HTTPS (certificados SSL/TLS) |

| | |
|---|---|
| Identificador | RNF-3 |
| Nombre | Seguridad perimetral del servidor |
| Descripción | El servidor debe tener los elementos y sistemas necesarios para denegar el acceso al mismo a personas no autorizadas. Se hará uso de herramientas como firewalls, pentesting… |

| | |
|---|---|
| Identificador | RNF-4 |
| Nombre | Copia de seguridad |
| Descripción | Se deberán realizar copias de seguridad automáticas de las configuraciones del servidor y de las bases de datos para minimizar el impacto en caso de pérdida de datos. |

| | |
|---|---|
| Identificador | RNF-5 |
| Nombre | Disponibilidad |

| | |
|---|---|
| Descripción | El sistema debe estar activo y disponible para los usuarios en todo momento. Una estimación general permitiría unos tiempos muertos programados para mantenimiento preventivo de 30 minutos por semana. Asimismo, una estimación general de unos tiempos muertos NO programados debido a fallos y ajustes de 3 horas anuales. |

| | |
|---|---|
| Identificador | RNF-6 |
| Nombre | Usabilidad |
| Descripción | Las interfaces deben ser intuitivas de manera que se requiera una intervención mínima por parte del usuario para la realización de las principales funcionalidades. |

| | |
|---|---|
| Identificador | RNF-7 |
| Nombre | Escalabilidad |
| Descripción | La infraestructura debe posibilitar la sencilla escalabilidad del sistema. |

| | |
|---|---|
| Identificador | RNF-8 |
| Nombre | Fiabilidad |
| Descripción | La plataforma debe transmitir confianza manteniendo la fiabilidad del sistema. |

## 7.3 Otros requisitos

| | |
|---|---|
| Identificador | OR-1 |
| Nombre | Auditoría |
| Descripción | Proveer un mecanismo de auditoría en forma de logs históricos para reflejar comportamientos sospechosos de los usuarios y agentes externos, además del posible mal funcionamiento del sistema. |

| | |
|---|---|
| Identificador | OR-2 |
| Nombre | Alineación con la legalidad |
| Descripción | Implementar las condiciones necesarias para cumplir con la RGPD (en inglés GDPR): reglamento relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos. |

## 7.4 Diagrama de casos de uso

Para visualizar los requisitos funcionales anteriores en relación a los actores que los realizan, se proponen los siguientes diagramas de uso de casos. Los principales actores que intervienen son el Paciente y el Profesional, pero éstos parten de ser un Usuario no Identificado.

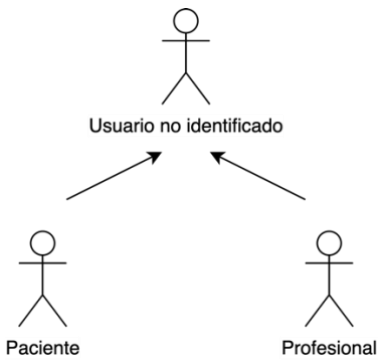


*Figura 5. Diagrama de caso de uso 1, usuarios*
*(Fuente Propia)*

Las únicas acciones que puede realizar este último actor son iniciar sesión de las dos modalidades disponibles. Iniciar sesión con doble factor de autenticación es obligatorio solo para el perfil de psicólogo. Una vez iniciada la sesión, el actor se convierte en paciente o profesional, dependiendo de su perfil.


*Figura 6. Diagrama de caso de uso 2, autenticación*
*(Fuente Propia)*

El paciente puede realizar las siguientes acciones:

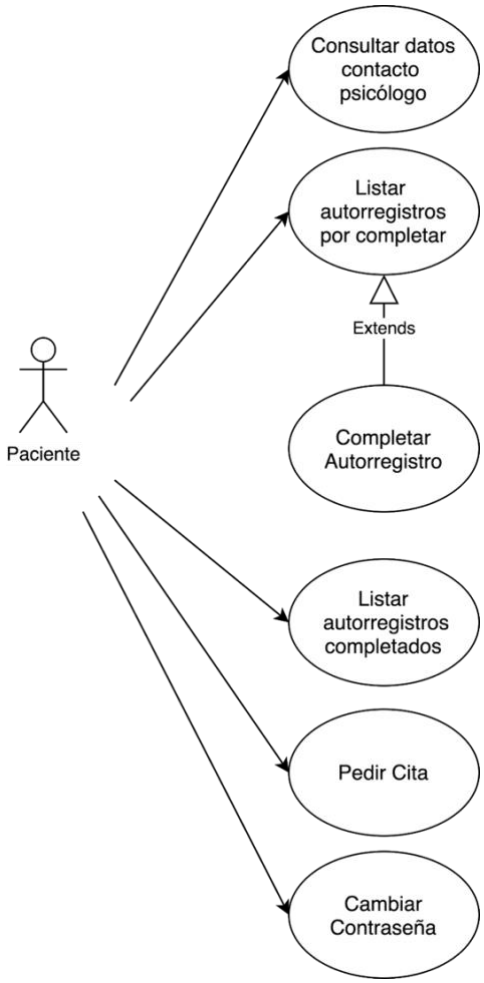


*Figura 7. Diagrama de caso de uso 3, acciones del paciente*
*(Fuente Propia)*

Por último, el profesional tiene acceso a las siguientes funcionalidades:

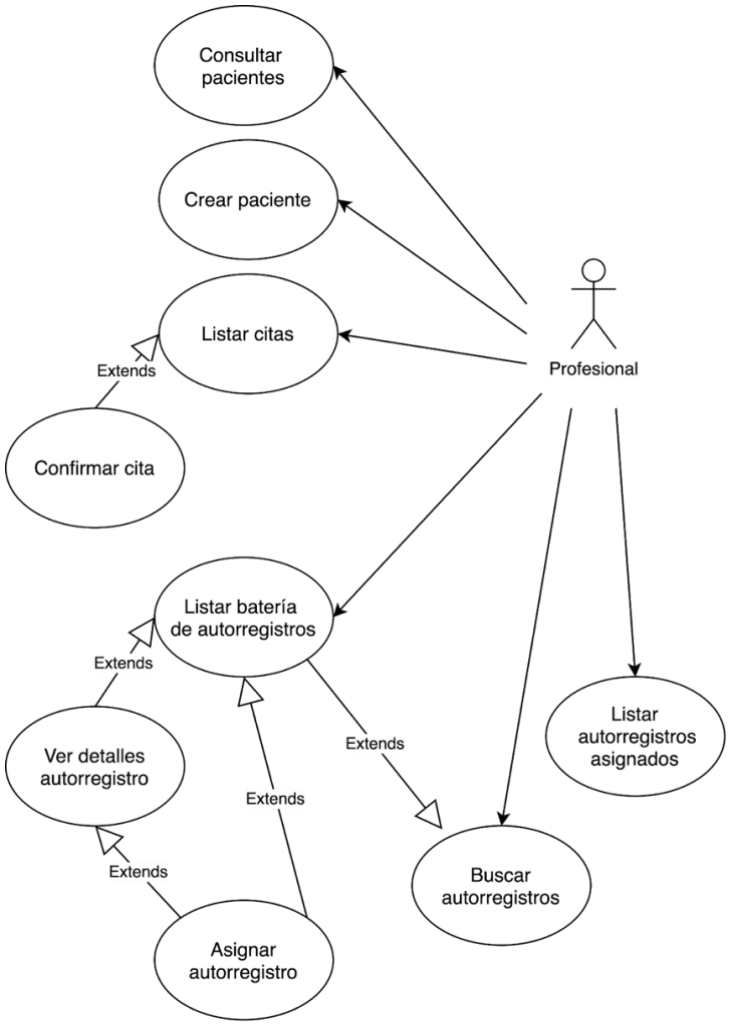


*Figura 8. Diagrama de caso de uso 4, acciones del psicólogo (Fuente Propia)*

# 8. Diseño

En este capítulo se abordará el diseño de la arquitectura técnica del trabajo, así como las soluciones a la problemática descrita en los capítulos iniciales. Asimismo, se dará respuesta a los requisitos definidos en el apartado de Análisis y Especificación.

Cabe destacar que este capítulo se enfocará especialmente a la aplicación a desarrollar, ya que gran parte de los elementos del proyecto global ya vienen facilitados por la empresa responsable del mismo.

## 8.1 Diseño de la arquitectura conceptual

Antes de poner el enfoque sobre la aplicación, hay que reconocer que ésta forma parte de un proyecto más grande, el cual cuenta con varios elementos y entidades. Teniendo esto en cuenta, en la *Figura 9. Diseño de la arquitectura del proyecto* se puede observar el diseño general de la arquitectura del mismo.

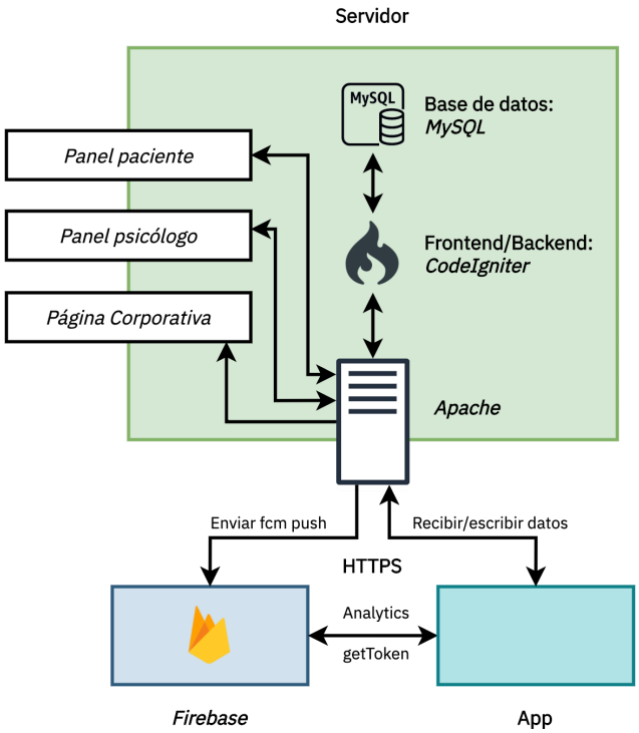


*Figura 9. Diseño de la arquitectura del proyecto*
*(Fuente Propia)*

Las tres grandes entidades que podemos destacar son:

- ***Firebase***, un elemento externo. Se utiliza principalmente para *cloud messaging [13]*, que permite habilitar las notificaciones push de manera sencilla para aplicaciones móviles, independientemente de la plataforma.

  A través de la conexión con la aplicación, se le otorga un token a la misma, el cual posteriormente permite al servidor hacer las peticiones necesarias para enviar las notificaciones.

- El ***servidor***, es el elemento propio más robusto. Incluye:
  - La API, principalmente desarrollada con CodeIgniter. Esta tecnología se escogió durante el desarrollo inicial del servicio web (2015), pero hay planes para migrar a Laravel, un framework php más actual.
  - La base de datos con MySQL.

El servidor se encarga de nutrir de datos a la aplicación y a los paneles de los diferentes perfiles de usuarios del servicio. Además, se encarga de enviar las notificaciones push a los dispositivos de los pacientes a través de *Firebase*.

- La ***aplicación***, el nuevo elemento que aparece en el esquema. Ésta se comunica bidireccionalmente tanto con el *servidor* como con *Firebase.* Del servidor se recogen, consultan y envían datos. De firebase se recoge un token para el posterior uso en la mensajería push.

Cabe mencionar que en todos los casos se utiliza un canal de comunicación seguro (HTTPS) y que la aplicación consume la API haciendo uso de un token de aplicación y un token de usuario.

## 8.2 Diseño de la arquitectura tecnológica

Este apartado contempla las tecnologías que son utilizadas para el desarrollo de la aplicación, las cuales se han escogido en función de los requerimientos, tanto funcionales como no funcionales.

### 8.2.1 Tecnologías del proyecto actual

Antes de mencionar las tecnologías a utilizar en la aplicación, se procede a listar las tecnologías utilizadas por el servicio en general.

Para el frontend del panel web de psicólogos y pacientes y la página corporativa, se destaca el uso de JQuery, Boostrap, además de las tecnologías básicas como html 5, css 3 y javascript. JQuery, Boostrap eran las herramientas más utilizadas cuando el proyecto fue desarrollado.

El encargado de gestionar y proporcionar las páginas web es CodeIgniter. Esta herramienta, además, sirve como backend y habilita la API utilizada tanto por los servicios web como por la aplicación. CodeIgniter era el framework más popular cuando el proyecto fue concebido, pero tiene muchas similaridades con Laravel, funcionando ambos de manera parecida y estando basados en PHP.

A pesar de utilizar tecnologías con varios años de antigüedad, el proyecto ha sido actualizado para ser completamente funcional y seguro. Aun así, la empresa tiene planes de migrar el proyecto a una tecnología más reciente que pueda facilitar el mantenimiento y pueda adaptarse mejor a las nuevas tecnologías y especificaciones. Una de las opciones mejor posicionadas es Laravel, ya que la empresa trabaja con este nuevo framework desde hace años.

Para el frontend, se actualizarán las páginas con un diseño más moderno y actual, y se mejorará la experiencia del usuario para ofrecer el mejor servicio posible.

En cuanto a la base de datos, tal como se ha comentado en el apartado anterior, se utiliza MySQL como motor y, además, funciona de manera esperada, cumpliendo con los requerimientos de estabilidad, rendimiento y seguridad.

### 8.2.2 Tecnologías de la aplicación

Uno de los requisitos con mayor peso es el desarrollo de una aplicación multiplataforma (Android e iOS). Con esto en mente, se ha descartado la realización de dos aplicaciones nativas, siendo inviable principalmente por el tiempo que ese desarrollo tomaría.

Se han investigado los principales frameworks para el desarrollo de aplicaciones multiplataforma, y la empresa ha decidido que la aplicación a desarrollar para este trabajo final de máster sea construida con Ionic [1] y Angular [2].

A pesar de que tecnologías como Flutter [3] o React Native [4] hayan ganado más peso en la actualidad, se ha decidido la combinación de Ionic y Angular por la familiaridad que tiene para el equipo de la empresa.

Las demás aplicaciones móviles de la empresa han sido también desarrolladas con Ionic e Angular, además de que esta combinación de tecnologías permitiría un mantenimiento más sencillo y económico.

Además, el uso de Angular también se justifica por la principal actividad de la empresa, que es el desarrollo web. Esta tecnología facilitaría la creación de nuevas páginas y funcionalidades.

Para completar las funcionalidades de la aplicación y cumplir con los requisitos y las especificaciones definidas, se procede a usar plugins de Ionic y Cordova. Éstos se mencionan en detalle en los siguientes apartados.

## 8.3 Diseño de la persistencia

En este apartado se elaborará el diseño del sistema que hará que los datos manejados por el usuario sean persistentes y accesibles en la aplicación.

### 8.3.1 Persistencia del servidor

El servicio parte de tener un sistema de bases de datos funcional, el cual sustenta las aplicaciones web del mismo. Esta base de datos es un entramado bastante complejo de entidades, por lo que no ha sido proporcionado por la empresa para su exposición en este trabajo de final de máster.

Sin entrar en detalles, las entidades más importantes son los usuarios, los tipos de preguntas de autorregistros, los autorregistros (formularios) completos, las configuraciones de los paneles, el carrito, los pedidos y las traducciones a los distintos idiomas en los que está disponible el servicio.

Por último, a grandes rasgos se puede comentar que tiene MySQL como motor, por lo que se trata de una base de datos relacional. Además de las configuraciones estándar de seguridad, tiene un sistema de copias diarias.

### 8.3.2 Persistencia de la aplicación

Tras plantear la pregunta sobre qué datos se van a almacenar en la aplicación, se ha determinado que solamente serán los datos básicos del perfil del usuario, el token para realizar peticiones a la API de manera segura y una serie de variables necesarias para el funcionamiento correcto de la aplicación.

Por decisión de la empresa, los autorregistros y las respuestas del paciente no se almacenarán en el dispositivo, sino que estarán solamente disponibles cuando el paciente utilice la aplicación, almacenadas en variables de sesión. Esto implica que el paciente debe de tener una conexión a internet para consultar los datos internamente haciendo una petición al servidor para recibirlos.

En cuanto a la herramienta que se utilizará para guardar los datos, Ionic ofrece varias posibilidades:

- **Ionic Offline Storage** [5], un plugin premium mantenido de manera oficial por el equipo de Ionic. Depende del uso que se le vaya a dar, utiliza un motor NoSQL o uno SQL (SQLite) y destaca por su gran rendimiento.

  Además, se trata de la solución más robusta y segura, ya que cifra la base de datos con el estándar de cifrado AES y almacena la clave de manera segura. Por último, ofrece un servicio de *cloud* para un sencillo almacenamiento de los datos en la nube.

- **Ionic Storage** es una alternativa open source mantenida también por Ionic. A diferencia de la solución anterior, Ionic Storage no ofrece un cifrado de los datos, sino que guarda los datos en claro.

  Se encarga de guardar los datos en pares clave / valor utilizando principalmente SQLite como motor. Destaca por ser muy estable y quitarle la responsabilidad a los sistemas operativos de gestionar los datos, evitando así casos de pérdida de datos en situaciones de poco almacenamiento en los dispositivos.

- **Secure Storage** [6] es la variante segura de Ionic Storage, que utiliza distintos métodos para cifrar los datos. Este plugin ha sufrido varios altibajos, especialmente en un tiempo reciente al quedarse sin soporte, pero el proyecto continúa vivo gracias a la comunidad.

  La forma de guardar los datos es muy similar a la de Ionic Storage, pero los métodos que utiliza para cifrar o guardar los datos de manera segura los datos son:
  - En iOS, se almacenan los datos en el KeyChain del dispositivo, utilizando la librería de SAMKeychain. Este método tiene la ventaja de utilizar el cifrado nativo de los dispositivos iOS.
  - En Android, al no tener un sistema nativo como el KeyChain de iOS, los datos se cifran con AES, utilizando una clave generada aleatoriamente. Esta clave se cifra con un par de claves RSA del Android KeyStore del dispositivo. La clave AES cifrada es almacenada en SharedPreferences.

  Este método tiene bastantes desventajas, ya que es necesario que el usuario tenga un bloqueo de pantalla activo para la creación de la base de datos. Además, cada vez que

se cambia el método de bloqueo, la KeyStore se restablece, lo que inutiliza la base de datos anterior pues no se puede acceder a los datos.

En un principio se pensó en utilizar Ionic Offline Storage, pero la idea fue descartada debido a la falta de documentación y el coste elevado que supondría el pago por su uso. Ionic Storage resulta muy insegura y no es la opción más adecuada para esta aplicación, la cual maneja varios datos sensibles.

A pesar de sus desventajas, se procederá a utilizar Secure Storage como herramienta de almacenamiento principal. Además, por decisión de la empresa, se utilizará un sistema compuesto tanto por Secure Storage como por Ionic Storage.

Aunque se le dé prioridad a Secure Storage, se permitirá el almacenamiento de los datos sensibles en Ionic Storage si este no tiene un método de bloqueo del dispositivo. Se mostrará un mensaje informativo si la aplicación no detecta que no existe un método de bloqueo y se facilitarán un botón para activar un método de bloqueo en la configuración del dispositivo y otro para usar Ionic Storage, siendo así su uso una decisión explícita del usuario.

## 8.4 Diseño de la seguridad

Al tratar con datos sensibles es inevitable pensar en el diseño de una capa de seguridad. Además, este paso resulta ser imprescindible por la necesidad de adecuarse a la legalidad y cumplir con el nuevo Reglamento General de Protección de Datos (RGPD).

Por parte de la aplicación, las medidas de seguridad a tomar son sencillas. Como ya se ha mencionado con anterioridad, los únicos datos que se almacenan del paciente son su identificador dentro del sistema, su alias y un token de usuario para la comunicación con la API.

Estos datos son almacenados, a no ser que el usuario escoja utilizar la aplicación sin esta opción, haciendo uso del Secure Storage de Ionic. De ser usado, obliga al usuario a tener un bloqueo de

pantalla seguro, lo que mejora la seguridad del dispositivo y se convierte en una primera barrera para posibles terceros que intenten acceder a la información del paciente.

En cuanto a los datos del usuario, el sistema no requiere del paciente ningún dato personal sensible como puede ser un número de teléfono o un correo electrónico, ya que su cuenta es generada por el psicólogo en el panel correspondiente. Para identificarse al sistema, se le genera un alias y una contraseña aleatorias, además de un identificador interno para facilitar la gestión de los datos en la base de datos.

Dentro de la aplicación se habilita una pantalla específica para cambiar esta contraseña generada si lo desea.

Además de todas estas medidas, se desarrollará también un sistema de bloqueo de la aplicación, por petición de la empresa. Este sistema servirá como una segunda capa de seguridad y bloqueará la aplicación siempre que el usuario salga de la misma, ya sea cerrándola por completo o enviándola a segundo plano.

Se le permitirá al usuario desbloquear la aplicación mediante:

- Un PIN de 4 dígitos. Para lograr este método, se desarrollará desde cero, ya que los plugins disponibles no cumplen los requisitos del proyecto.
- Biometría, huella o reconocimiento facial. Se usará alguno de los plugins disponibles para traer esta funcionalidad.
- Alias y contraseña, el método más costoso en cuanto a tiempo, pero servirá también como método por defecto en el caso que se tenga activo otro y no se recuerde o pueda acceder con el mismo.

Por último, cabe mencionar que la comunicación con todas las entidades externas (servidor o Firebase) se realizan a través de un canal seguro (HTTPS).

## 8.5 Diseño de la API

La API del proyecto viene completamente proporcionada por la empresa y el alumno no ha contribuido en el desarrollo de la misma, siendo la única parte referente a la aplicación de la cual el alumno no es el autor. A pesar de esto, se incluye en este documento a modo de referencia para definir los servicios a los que se deberá conectar la aplicación.

Como ya se ha mencionado en el apartado del diseño de la arquitectura tecnológica, la API ha sido desarrollada con CodeIgniter. Esta herramienta se encarga de gestionar tanto el backend como el frontend, por lo que resulta una pieza fundamental de la arquitectura de este proyecto.

Se procede a listar en forma de tablas los endpoints utilizados en la aplicación desarrollada. Las tablas están formadas por:

- **Definición:** una breve descripción de la llamada.
- **Endpoint**: la ruta específica de la llamada correspondiente.
- **Método**: el método http de la llamada.
- **Parámetros**: tendrá valor si se envían datos en la llamada.
- **Resultado**: el comportamiento o datos esperados como resultado de la llamada.

Los endpoints son:

| | |
|---|---|
| Definición | Hacer login en el sistema. |
| Endpoint | /users/login |
| Método | POST |
| Parámetros | El alias del usuario, su contraseña y el token de la aplicación. |

| | |
|---|---|
| Resultado | En caso de autenticar, se devuelve un objeto con un token con una validez de 30 días, el identificador del usuario y el identificador del psicólogo / consultorio al que acude.<br><br>Un error en caso contrario. |

| | |
|---|---|
| Definición | Obtener los autorregistros de un paciente. |
| Endpoint | /autorregistros/autoUser |
| Método | POST |
| Parámetros | El identificador del paciente y su token de usuario válido. |
| Resultado | En el caso de ser los parámetros correctos, se devuelve un objeto con todos los autorregistros del paciente. Estos autorregistros tienen un nombre, un identificador, una descripción, un booleano que representa si está activo y unas fechas de realización.<br><br>Cabe destacar que esta petición no trae las preguntas de los autorregistros. Para ello se utilizará otra llamada.<br><br>Se devuelve un error en caso de no ser los parámetros correctos. |

| | |
|---|---|
| Definición | Obtener las preguntas de un autorregistro. |
| Endpoint | /autorregistros/autoQuestions |
| Método | POST |
| Parámetros | El identificador del paciente, su token de usuario válido y el identificador del autorregistro. |

| | |
|---|---|
| Resultado | En el caso de ser los parámetros correctos, se devuelve un objeto con todas las preguntas del autorregistro en cuestión. Estas preguntas tienen un título, instrucciones de cómo responder, el tipo de pregunta (que cambiará el tipo input dentro de la aplicación) y un identificador.<br><br>Se devuelve un error en caso contrario. |

| | |
|---|---|
| Definición | Responder a un autorregistro. |
| Endpoint | /autorregistros |
| Método | POST |
| Parámetros | El identificador del paciente, su token de usuario válido, el identificador del autorregistro y todas las respuestas encapsuladas en un array. |
| Resultado | En el caso de ser todos los parámetros correctos, se responde el autorregistro correspondiente en la base de datos y se devuelve una confirmación.<br><br>Se devuelve un error en caso contrario. |

| | |
|---|---|
| Definición | Pedir cita (como paciente). |
| Endpoint | /citas/nueva |
| Método | POST |
| Parámetros | El identificador del paciente, su token de usuario válido y una fecha y una hora válidos. |

| | |
|---|---|
| Resultado | En el caso de enviar parámetros correctos, se creará la cita en espera y se devolverá una confirmación y un identificador de cita. Se recuerda que las citas tienen que ser aceptadas y confirmadas por el psicólogo antes de notificar al paciente que su cita está aprobada.<br><br>Se devuelve un error en caso contrario. |

| | |
|---|---|
| Definición | Consultar los datos de una cita en espera. |
| Endpoint | /citas/consultar |
| Método | POST |
| Parámetros | El identificador del paciente, su token de usuario válido y el identificador de la cita |
| Resultado | En el caso de ser todos los parámetros correctos, se devuelve la información de la cita en cuestión. La cita tiene la fecha, la hora y el estado (en espera, denegada o aprobada).<br><br>Se devuelve un error en caso contrario. |

| | |
|---|---|
| Definición | Cambiar la contraseña del paciente (acción realizada por el mismo paciente). |
| Endpoint | /users/cambiarPass |
| Método | POST |
| Parámetros | El identificador del paciente, su alias, su token de usuario válido y la nueva contraseña. |
| Resultado | En el caso de ser todos los parámetros correctos, se cambia en la base de datos la contraseña y se devuelve una confirmación. |

| | |
|---|---|
| | Se devuelve un error en caso contrario. |

| | |
|---|---|
| Definición | Añadir el token de firebase de la instalación del paciente para poder recibir notificaciones push. |
| Endpoint | /users/addFirebaseToken |
| Método | POST |
| Parámetros | El identificador del paciente, su token de usuario válido y el token de firebase. |
| Resultado | En el caso de ser todos los parámetros correctos, se añade en la base el token de firebase enviado y se devuelve una confirmación.<br>Se devuelve un error en caso contrario. |

| | |
|---|---|
| Definición | Consultar los datos del psicólogo y el consultorio asignado. |
| Endpoint | /consultorio/info |
| Método | POST |
| Parámetros | El identificador del paciente, su token de usuario válido y el identificador del psicólogo / consultorio. |
| Resultado | En el caso de ser todos los parámetros correctos, se devuelve un objeto con los datos del psicólogo / consultorio que se ha pedido. Este objeto contiene el nombre del consultorio, el psicólogo asignado al paciente, la dirección del consultorio, las coordenadas del mismo y un correo electrónico de contacto.<br>Se devuelve un error en caso contrario. |

## 8.6 Diseño de interacción e interfaces

Habiendo resuelto la parte más técnica de la aplicación, se procede a analizar las pantallas con las cuales interaccionará el usuario para llevar a cabo distintas actividades.

Las interfaces principales de la aplicación y el estilo de las mismas han sido facilitadas por el departamento de diseño de la empresa, los cuales han construido una serie de pantallas con un diseño moderno y simple. Se destaca el uso del color característico del servicio y una serie de patrones que se comentarán en el siguiente apartado, Guía de Estilos.

La herramienta que se ha utilizado para la elaboración de estas interfaces es Abode XD, bastante utilizada en el prototipado y diseño de todo tipo de interfaces de la empresa. Adobe XD, a pesar de tener como competencia gigantes como Sketch y Figma, se integra perfectamente con el ecosistema de Adobe, lo que gusta a la empresa por ser la suite de aplicaciones utilizada por el departamento de diseño.

Al entrar a la aplicación, la primera pantalla que se encuentra el usuario es el login. Esta interfaz se puede apreciar en la *Figura 10. Diseño de la pantalla de Login.*

Aquí el usuario no tiene demasiadas opciones, ya que sólamente puede acceder a su cuenta introduciendo su alias y contraseña o recuperar la contraseña pulsando el botón indicado.

De entrada ya se tiene una idea del diseño general que va a seguir la aplicación, especialmente el color, presente en los círculos superior e inferior. Estos círculos se procederán a animar.

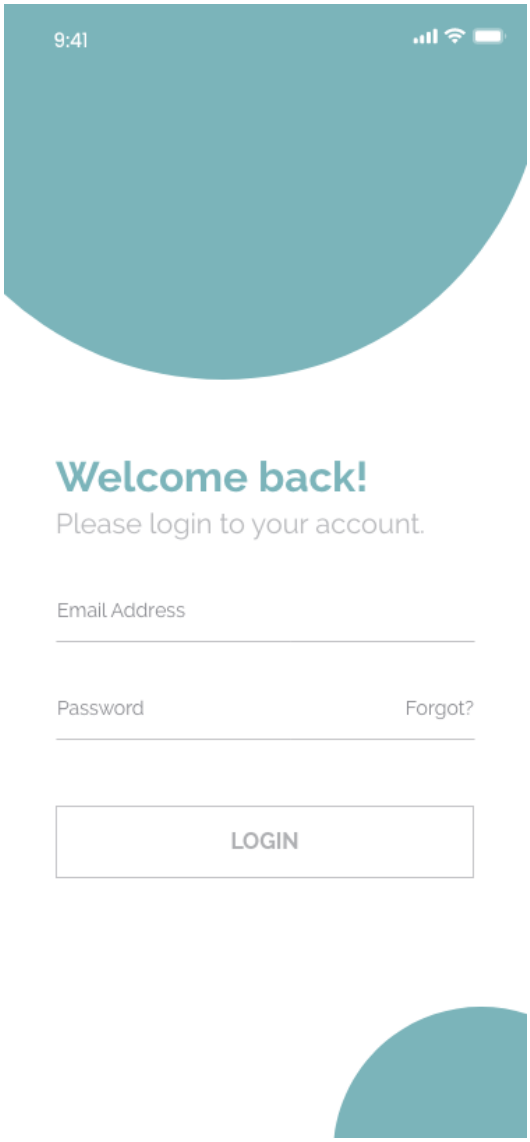


*Figura 10. Diseño de la pantalla de Login (Pixel78)*

La pantalla que más verá el usuario es la pantalla de formularios (autorregistros). Las acciones que podrá realizar el usuario serán abrir el menú izquierdo pinchando en el botón superior izquierdo, consultar los formularios por hacer y los ya completados y por último, entrar en los mismos para consultar sus respuestas o responder.

Esta pantalla se puede apreciar en la *Figura 11. Diseño de la pantalla de Listado de Autorregistros*. Se prescindirá de la barra de búsqueda para el perfil de paciente y solamente estará disponible para el perfil de psicólogo.


*Figura 11. Diseño de la pantalla de Listado de Autorregistros (Pixel78)*

Para separar los autorregistros por completar y los ya completados, se utilizará un slider, lo cual permitirá navegar entre las dos diapositivas mediante un gesto swipe.

La segunda pantalla más importante de la aplicación es la pantalla de completar autorregistros. Se accede a la misma desde la pantalla de formularios y construirá las preguntas según el tipo de las mismas. Se tendrá una serie de componentes que representen todos los tipos posibles de

preguntas y se manejará un formulario dinámico de angular. Además de esto, se mostrarán las instrucciones que el paciente debe seguir para completarlo.

Esta pantalla se puede observar en la *Figura 12. Diseño de la pantalla de Completar Autorregistro.*


*Figura 12. Diseño de la pantalla de Completar Autorregistro (Pixel78)*

La aplicación tendrá también un menú lateral, situado en la parte izquierda. Será accesible únicamente desde la pantalla de formularios y dará acceso a las demás pantallas de la aplicación.

Su diseño es bastante minimalista y contiene únicamente los enlaces necesarios y un botón de cerrar. Este menú se puede apreciar en la *Figura 13. Diseño del Menú Lateral.*

Las principales opciones serán:

- Desconexión del perfil del usuario, que borrará todos los datos guardados por la aplicación y devolverá al usuario a la pantalla de login.
- Acceso a la pantalla de contacto, donde podrá consultar los datos más relevantes de contacto del consultorio al que acude, además de poder abrir la ubicación del mismo en la aplicación de mapas deseada.
- Acceso a la pantalla de pedir cita, donde podrá consultar los datos de la próxima cita o solicitar una si no tiene ninguna.
- Acceso a la pantalla de cambiar contraseña, la cual permitirá al paciente cambiar su contraseña y proteger su cuenta.
- Acceso a la pantalla de cambio de método de bloqueo de la aplicación, la cual guiará al usuario en este proceso y podrá activar cualquiera de las opciones disponibles.


*Figura 13. Diseño del Menú Lateral (Pixel78)*

La pantalla de cambiar contraseña se puede apreciar en la *Figura 14. Diseño de la pantalla de Cambiar Contraseña*. En esta pantalla únicamente podrá introducir su contraseña actual, una nueva contraseña y repetir esta última. Se mostrará una confirmación si todo ha funcionado correctamente.

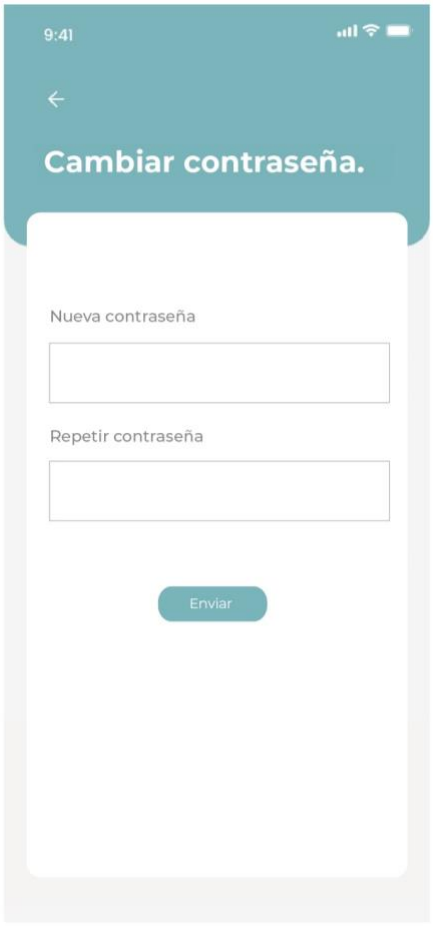


*Figura 14. Diseño de la pantalla de Cambiar Contraseña (Pixel78)*

Otra pantalla interesante es la pantalla de confirmación de login, donde el psicólogo podrá introducir un código de doble factor para acceder a la aplicación. Esta pantalla está representada en la *Figura 15. Diseño del acceso por Doble Factor de Autenticación.*

A causa del escaso tiempo disponible, es probable que el perfil de psicólogo no se llegue a implementar para esta primera versión, pero servirá de inspiración para otras pantallas que no están previstas en el primer diseño de la aplicación.


*Figura 15. Diseño del acceso por Doble Factor de Autenticación (Pixel78)*

Además de las principales interfaces, se presentan los modales que se usarán para dar información sobre el resultado de una acción o simplemente mostrar un aviso o advertencia.

Estos modales siguen la misma estética vista en las interfaces anteriores y completan favorablemente el diseño de la aplicación. Se pueden consultar en la *Figura 16. Diseño de los modales personalizados.*

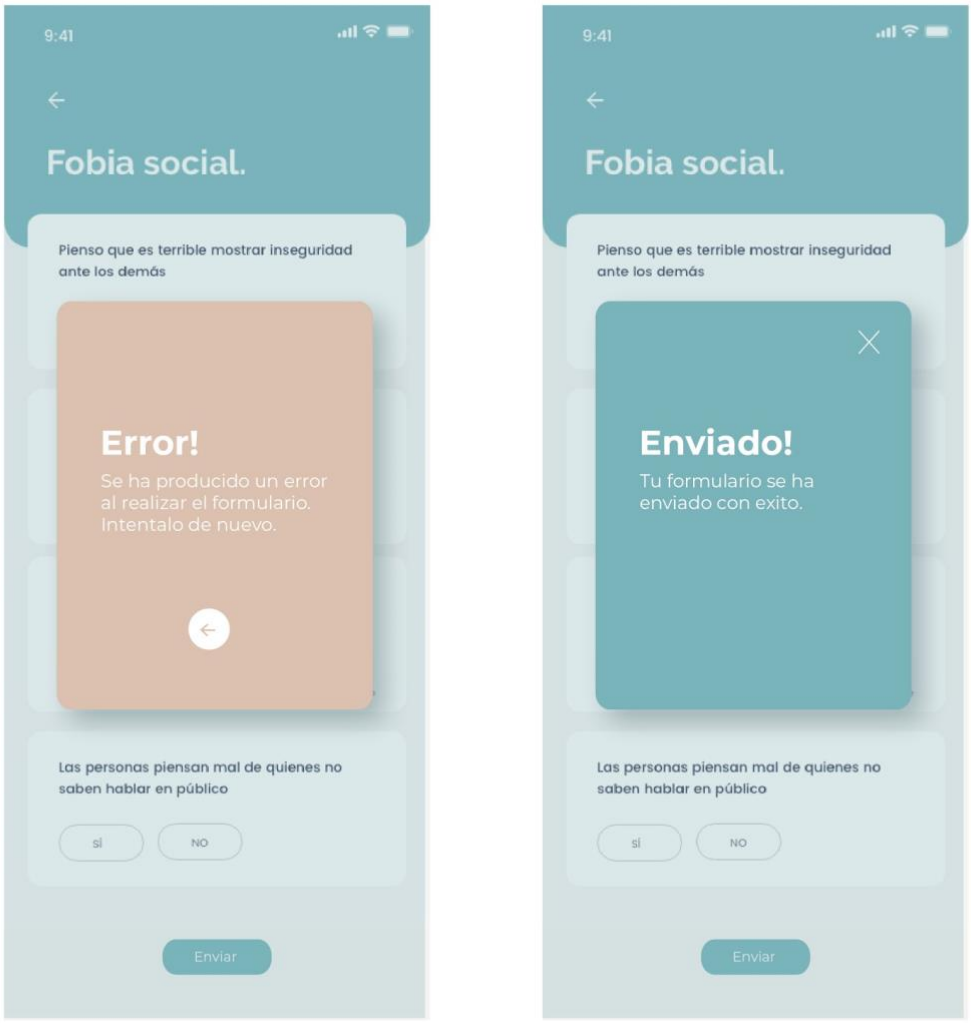


*Figura 16. Diseño de los modales personalizados*
*(Pixel78)*

Por último, la empresa también ha facilitado el diseño del splashscreen, pantalla que será vista en la carga inicial de la aplicación y contendrá el logo del proyecto y el color principal del mismo. Además, también se incluye un pequeño modal que aparecerá cuando se acceda a la pantalla de formularios y mostrará tarjetas con los últimos autorregistros asignados por el psicólogo. Estás pantallas se pueden observar en las figuras *Figura 17. Diseño del SplashScreen* y *Figura 18. Diseño del modal informativo, autorregistros por completar*.


*Figura 17. Diseño del SplashScreen (Pixel78)*


*Figura 18. Diseño del modal informativo, autorregistros por completar (Pixel78)*

Las pantallas restantes estarán inspiradas en las ya presentadas y compartirán los mismos patrones de diseño.

## 8.7 Guía de estilos

El diseño de la aplicación se ha podido observar a lo largo de la presentación de las principales interfaces de la aplicación. Se trata de un diseño moderno con un estilo muy marcado y unos elementos característicos que se comparten en todas las pantallas.

Se busca minimalismo y simplicidad en los elementos de las interfaces. Además, se juega con tarjetas y bordes redondeados para ofrecer un diseño más agradable y se utilizan las profundidades y las sombras para destacar los elementos más importantes en cada contexto de la aplicación.

La identidad del proyecto también se ve reflejada a través del logotipo y el isotipo diseñados por la empresa. El isotipo se utiliza como icono de la aplicación y se encuentra presente también en la pantalla de formularios. El logotipo se utiliza en el splashscreen de carga de la aplicación. Éstos se pueden apreciar en las siguientes figuras:

*Figura 19. Isotipo de Augister*
*(Pixel78)*

AUGISTER

*Figura 20. Logotipo de Augister*
*(Pixel78)*

Uno de los elementos más llamativos de las interfaces es su color principal, que otorga personalidad al diseño y aporta bastante a la identidad del proyecto. A continuación, se muestra un listado con los colores utilizados:

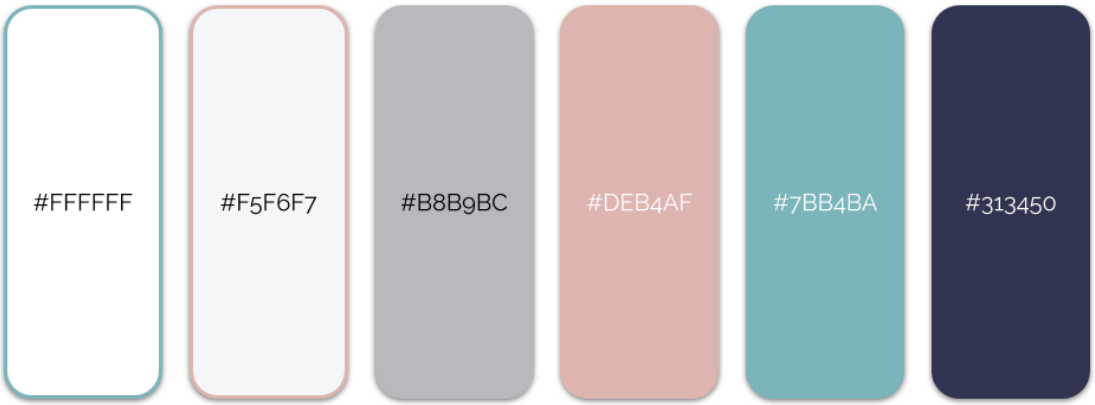


*Figura 21. Colores de Augister*
*(Fuente Propia)*

En cuanto a los usos de los mismos:

- #FFFFFF, #B8B9BC y #323450 se utilizan principalmente para textos. El uso de cada uno depende del contexto y de los colores del contenedor.
- #FFFFFF y #F5F6F7 y #7BB4BA para fondos de tarjetas o de la página.
- #7BB4BA es el color principal y también es usado para el modal de confirmación de que una operación se ha completado satisfactoriamente.
- #DEB4AF para el modal en situaciones de error.

Por último, la fuente utilizada es Raleway, disponible en Google Fonts. Raleway complementa el diseño y también otorga personalidad al mismo:

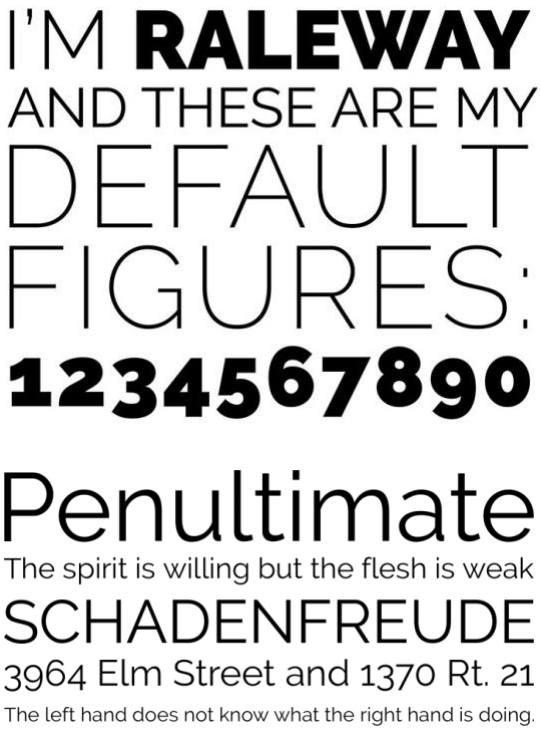


*Figura 22. Raleway, fuente de Augister (Google Fonts)*

## 8.8 Diseño de pruebas y validación

Para finalizar este capítulo, se procede a definir cómo se probará y validará la aplicación, tanto durante el proceso del desarrollo como al finalizar esta primera versión.

Al seguir una metodología de trabajo en el que hay un feedback de manera frecuente, después del desarrollo de cada funcionalidad, tanto el alumno como el equipo de la empresa probarán en detalle de manera manual que dicha funcionalidad se haya desarrollado de manera correcta y no altere el comportamiento de alguna otra funcionalidad de la aplicación.

De manera paralela, se elabora un guion a seguir para probar la aplicación de manera eficiente. Además, también se elabora un guion de usabilidad para comprobar los aspectos de usabilidad más importantes de la aplicación, de manera que los primeros usuarios de la aplicación puedan dar feedback y preparar una segunda versión mejorada.

Una vez completada la primera versión de la aplicación, volverá a ser probada en profundidad por el equipo de la empresa y pasará a ser probada por usuarios reales, los cuales recibirán el guion previamente mencionado y darán el feedback necesario para mejorar la versión de la aplicación existente.

Estos primeros usuarios serán los psicólogos y los pacientes del consultorio de uno de los responsables del proyecto, teniendo así un feedback real desde el principio. Esta información resultará muy valiosa para ajustar el comportamiento de la aplicación y ofrecer la mejor experiencia posible, así como corregir errores que se hayan escapado a la revisión final por parte del equipo de la empresa.

Es importante remarcar que, debido a la situación sanitaria actual, es probable que el testeo por parte de pacientes reales se demore y no entre en el tiempo del desarrollo y presentación de este trabajo final de máster. Aunque las pruebas y validaciones mencionadas se escapen del tiempo de entrega del trabajo, es un proceso que se realizará cuando sea posible para obtener un feedback valioso.

# 9. Implementación

Una vez se han definido todos los requisitos del sistema y se ha diseñado una solución viable, se procede a la implementación de la aplicación. La estructura de este capítulo ha sido organizada y ordenada siguiendo los mismos pasos que se han dado a lo largo del desarrollo.

Cabe destacar que también se incluyen tanto las dificultades que se han encontrado durante el desarrollo como las soluciones que se han dado.

## 9.1 Organización del proyecto

Como primer paso, se ha hecho una planificación de la organización del proyecto y se ha preparado el espacio de trabajo configurando las principales tecnologías a utilizar.

Como herramientas de desarrollo, se han usado el IDE Visual Studio Code, el terminal de macOS con los comandos de Ionic para generar páginas, componentes y ejecutar el proyecto en un host local para poder visualizarse en un navegador, el Node Package Manager como gestor de dependencias e instalación de plugins y Chrome y Safari como navegadores de desarrollo. En cuanto al gestor de versiones, se ha utilizado Git en combinación con un repositorio de BitBucket compartido con la empresa

Al tener el espacio de trabajo listo, se ha pasado a la creación del proyecto de la aplicación con Ionic 4 (la última versión estable en ese momento) y Angular, generando sus elementos. Cada uno de ellos forma parte de alguna de estas categorías:

- **Página**, una carpeta formada por:
    - Una vista HTML que contiene la parte visual de la página.
    - Un archivo de SCSS, la hoja de estilos de la página.
    - Un archivo Typescript que contiene la lógica de negocio de la página.

    - Un archivo módulo (Typescript) que contiene las importaciones de los elementos más importantes utilizados en la página.
    - Un archivo enrutador (Typescript) que contiene las rutas de la página.

- **Componente**, una carpeta que contiene todos los archivos de una página salvo el enrutador y el módulo. Funciona como un elemento dentro de una página.
- **Servicio**, un archivo Typescript que contiene la lógica de una entidad de la aplicación que se encarga de proporcionar datos o realizar una acción.

Partiendo de la carpeta src/app/ como raíz, se procede a listar los archivos y elementos más relevantes. Cabe destacar que esta lista contiene los archivos finales cuyas funcionalidades serán explicadas a lo largo de este capítulo.

- Init (página): página de inicio que se encarga de comprobar que el usuario está logueado y si existe una persistencia segura. Redirige a la página indicada en cada caso.
- Auth (carpeta): contiene la página de login, todos los componentes y páginas que intervienen en el sistema propio de bloqueo de la aplicación. También contiene las páginas de cambio de contraseña y confirmación del doble factor de autenticación del psicólogo.
- Paciente (carpeta): contiene las páginas de listado y completado de autorregistros, además de las páginas de citas y contacto.
- Registros (carpeta): contiene todos los componentes que representan cada uno de los once tipos de pregunta de autorregistro.
- Modales (carpeta): contiene los modales (componentes) customizados de error y éxito, los cuales se muestran dependiendo de la situación. Son llamados por las distintas páginas de la aplicación.
- Servicios (carpeta). Contiene los siguientes servicios:
    - DataService (Typescript): servicio que incluye el cliente http y las funciones con las llamadas a la API.
    - StorageService (Typescript): servicio que incluye la lectura y escritura en la persistencia local. Detecta qué tipo de persistencia ha escogido el usuario y adapta las funciones para utilizar la indicada, siendo totalmente transparente para los consumidores de este servicio.

    - TransitionService (Typescript): servicio que contiene las configuraciones del plugin de transiciones entre pantallas. Se definen una vez y se recuperan desde todas las páginas que lo necesiten.
    - LoadingService (Typescript): servicio que incluye un LoadingController de Ionic y su principal lógica. Se trata de un modal de carga.
- Shared (carpeta): contiene un archivo con constantes globales.

## 9.2 Diseño inicial y maquetación

Como siguiente paso se ha procedido a la maquetación de las principales pantallas y componentes. Se recuerda que las interfaces ya han sido facilitadas por la empresa.

El propósito de esta iteración ha sido crear una maqueta básica de la aplicación con el perfil del paciente, implementar el diseño facilitado por la empresa y tomar contacto con las tecnologías a utilizar a lo largo del desarrollo del proyecto. Además, es necesario remarcar que se parte de este punto sin tener acceso a la API del sistema, ya que se encontraba en una fase de migración de tecnologías.

Para mostrar datos en las vistas y adaptar cada interfaz a los datos que se recibirán de la API, se han utilizado variables con objetos javascript con datos randomizados, siguiendo el esquema proporcionado por la empresa.

El resultado de esta iteración han sido las pantallas que se proceden a definir a continuación.

### 9.2.1 Pantalla de login

El login se trata de la primera pantalla que se encuentra el usuario. Para el diseño de la misma se ha seguido al detalle la interfaz facilitada para la empresa. La *Figura 23. Pantalla de Login* refleja el resultado final de esta pantalla.

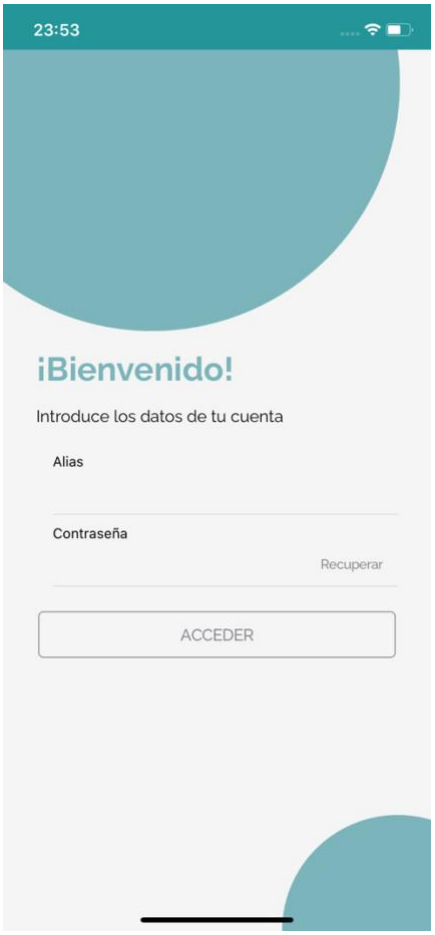


*Figura 23. Pantalla de Login*
*(Fuente propia)*

Como ya se adelantó en el capítulo de Diseño, esta pantalla es muy simple y el usuario puede realizar solamente 2 acciones, que son acceder a la aplicación autenticándose o recuperar la contraseña de su cuenta.

Entre los elementos más importantes que componen esta pantalla, los inputs del usuario se han conectado internamente con un formulario de Angular FormGroup, el cual puede gestionar de manera nativa restricciones y validaciones.

En cuanto a los círculos con el color principal de la aplicación, se han creado y animado totalmente con CSS. Para su posicionamiento se ha utilizado una posición fija, sin embargo, esto ha provocado problemas de superposiciones cuando el teclado está activo. Para solucionar este

problema y evitar que el diseño se rompa cada vez que aparece el teclado, el círculo inferior se oculta cuando este está activo.

### 9.2.2 Pantalla con listado de autorregistros

La segunda pantalla más importante es la que contiene el listado de autorregistros, tanto por completar como los ya completados. La *Figura 24. Pantalla de Listado de Autorregistros* muestra el resultado final de la misma.

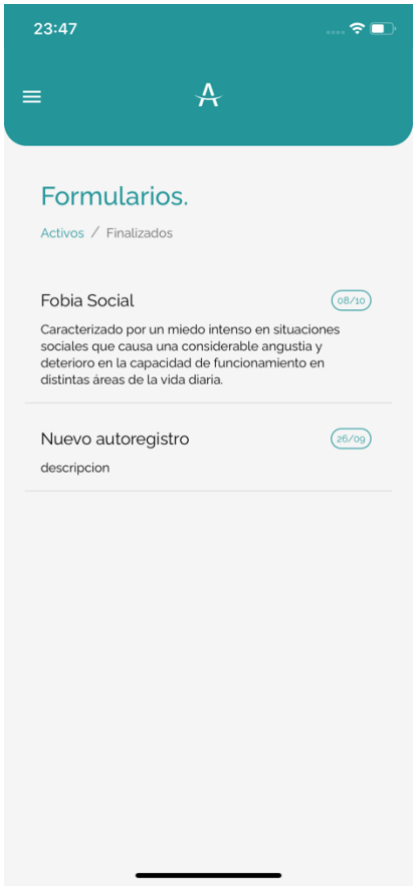


*Figura 24. Pantalla de Listado de Autorregistros*
*(Fuente propia)*

Los elementos más destacables son el contenedor de los autorregistros y el controlador que permite cambiar entre los diferentes tipos. Para el controlador se ha usado un SegmentController, adaptándolo al diseño del cliente.

Como contenedor de los autorregistros se ha usado un SlideController, que contiene dos páginas, una con cada tipo de autorregistros. Se ha dotado a este controlador con gestos de scroll y swipe:

- El swipe permite cambiar fácilmente entre los dos tipos de autorregistros sin tener que utilizar el controlador previamente mencionado.
- El evento de scroll se recoge y se utiliza para recoger los autorregistros del usuario de la API y actualizar la lista actual. Esta acción facilita la actualización de la lista actual de autorregistros, evitando que el usuario tenga que realizar esta acción de manera explícita.

### 9.2.3 Pantalla de autorregistro

La tercera interfaz más importante es la pantalla de autorregistro en la que el paciente completa las preguntas y envía las respuestas al psicólogo *(Figura 25. Pantalla de Autorregistro).*

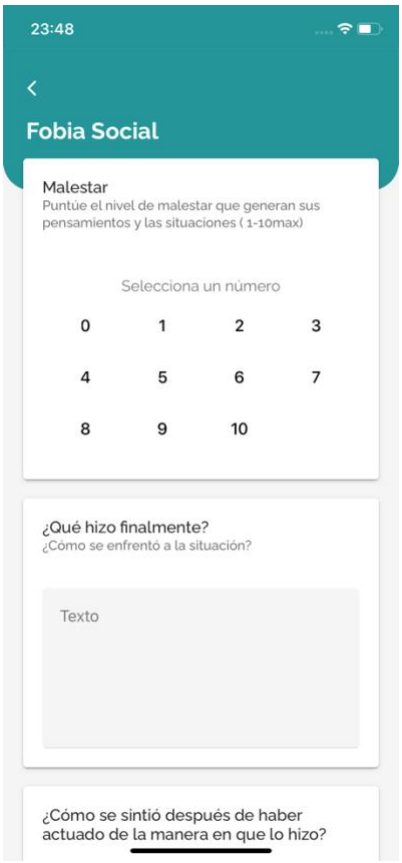


*Figura 25. Pantalla de Autorregistro*
*(Fuente propia)*

Esta vista es la que más complicaciones ha traído a la empresa, ya que la disposición de las preguntas del autorregistro puede influir enormemente en la usabilidad de esta interfaz.

Tras plantear varias opciones como un slider con una pregunta por página o unas tarjetas que se auto comprimen al completar el contenido, el resultado final es un listado vertical de tarjetas con una pregunta por tarjeta.

Esto obliga al usuario a hacer mucho scroll vertical si el autorregistro contiene muchas preguntas, pero permite tener un mejor acceso a las preguntas y evita una navegación excesiva en el caso de un slider y muchas páginas.

En cuanto a la estructura de la vista, se tiene una página padre y numerosos componentes individuales representando los once tipos de pregunta de autorregistro disponibles en el sistema. En la vista padre se tiene un controlador que recorre el objeto de preguntas y, según el tipo de las mismas, incrusta un componente u otro.

Los once tipos de pregunta son los siguientes:

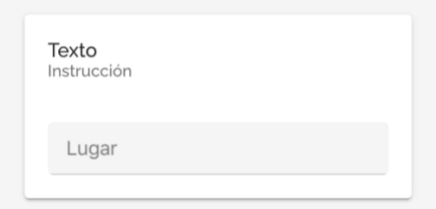


*Figura 26. Tipo de Pregunta: entrada de texto breve*

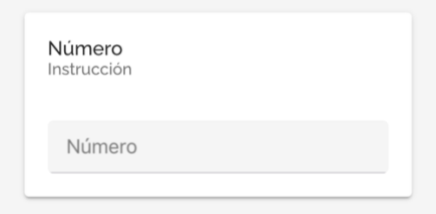


*Figura 27. Tipo de Pregunta: entrada de número*

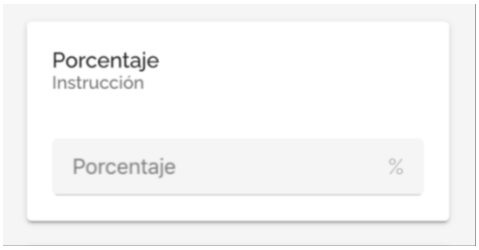


*Figura 28. Tipo de Pregunta: porcentaje, un valor entre 0 y 100*

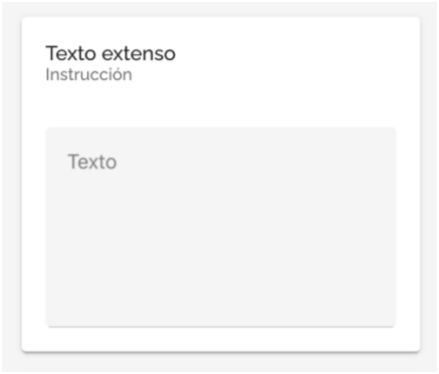


*Figura 29. Tipo de Pregunta: entrada de texto extenso*


*Figura 30. Tipo de Pregunta: seleccionar un valor entre 0 y 10*


*Figura 31. Tipo de Pregunta: seleccionar un valor entre 0 y 5*

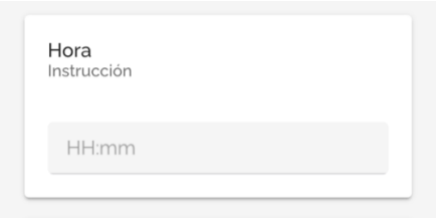


*Figura 32. Tipo de Pregunta: hora simple*


*Figura 33. Tipo de Pregunta: hora de inicio y fin*

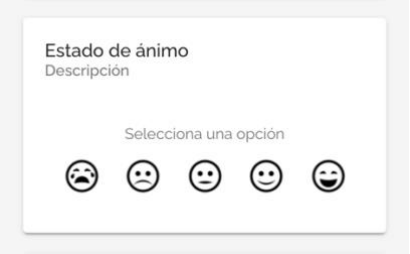


*Figura 34. Tipo de Pregunta: estado de ánimo*


*Figura 35. Tipo de Pregunta: entrada de texto y selección*

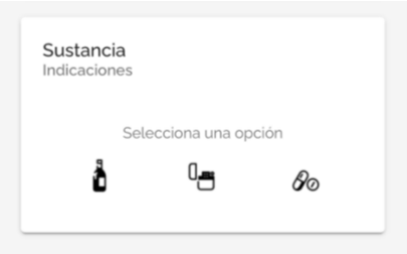


*Figura 36. Tipo de Pregunta: sustancia*

### 9.2.4 Otras pantallas

A continuación, se exponen las otras pantallas implementadas durante esta iteración.

La pantalla de contacto *(Figura 37. Pantalla de Datos de Contacto)* incluye la información básica y de contacto del consultorio al que acude el paciente como el nombre del lugar, el nombre del psicólogo personal, la dirección y un correo de contacto. Además, si el consultorio tiene unas coordenadas, se podrá abrir la aplicación de mapas del dispositivo en la ubicación del mismo.

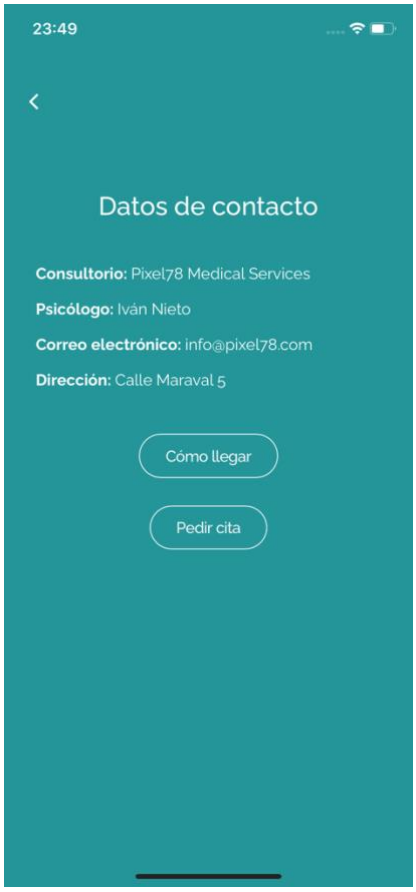


*Figura 37. Pantalla de Datos de Contacto*
*(Fuente propia)*

La pantalla de consultar cita / pedir cita *(Figura 38. Pantalla de Citas)* incluye las dos vistas relacionadas con pedir cita. Si el usuario no tiene ninguna cita se muestra una vista que incluye entradas de fecha y hora para poder solicitar la cita. En caso de tener alguna cita registrada, se mostrarán los datos de la misma con un indicador del estado de la misma y un botón para cancelarla.

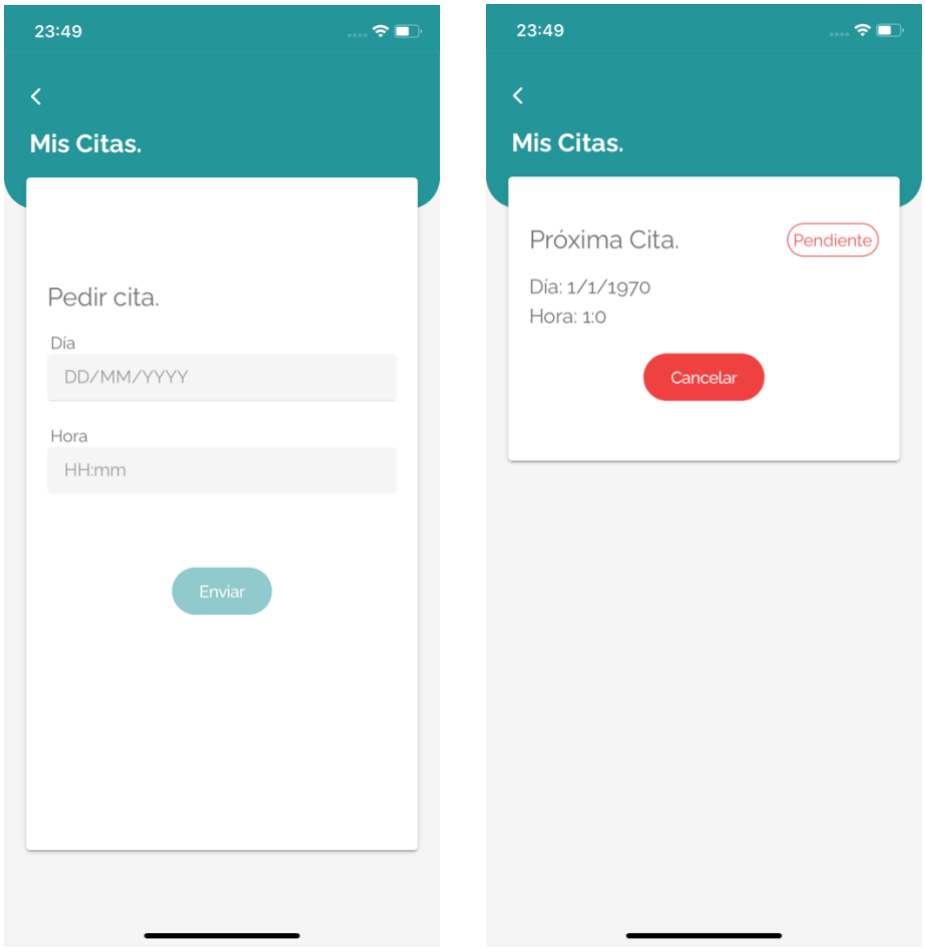


*Figura 38. Pantalla de Citas*
*(Fuente propia)*

La pantalla de cambiar contraseña *(Figura 39. Pantalla de Cambiar Contraseña)* incluye las entradas de texto necesarias para cambiar la contraseña del usuario. Estas entradas son la contraseña actual, una nueva contraseña y la repetición de la nueva.

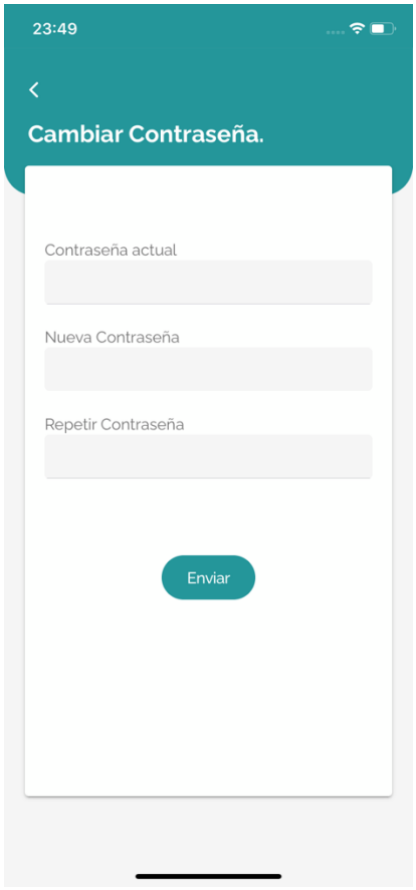


*Figura 39. Pantalla de Cambiar Contraseña*
*(Fuente propia)*

### 9.2.5 Estilos en Ionic

La principal actividad de esta iteración ha sido la maquetación y la aplicación de los estilos para tener unas interfaces lo más cercanas a los diseños facilitados por la empresa. Sin embargo, los estilos en ionic han resultado ser el problema más grande del desarrollo.

Ionic tiene numerosos elementos y componentes con un buen diseño y adaptados perfectamente al comportamiento de las dos grandes plataformas a las cuales se da soporte (Android e iOS).

Uno de los secretos de este framework es el uso de Shadow DOM para sus componentes.

HTML en un lenguaje que utiliza la estructura DOM en forma de árbol de nodos conectados que representan los diferentes elementos de un documento de marcado. Según la especificación de

la World Wide Web Consortium (W3C), Shadow DOM permite crear subestructuras ocultas cuyos elementos y código no pueden afectar a los nodos externos.

En el caso de Ionic, esto permite encapsular estructuras y evitar que los estilos de las mismas afecten otros elementos.

En Ionic existen dos estilos disponibles, Android (md) e iOS (ios) que, por defecto, se aplican según la plataforma en la que se esté ejecutando la aplicación. El uso de Shadow DOM permite que, de manera eficiente y controlada, se pueda cambiar y forzar un estilo diferente de la plataforma a un elemento en específico sin tener que cambiar el estilo global de la aplicación.

Sin embargo, a pesar de todas las ventajas que tiene el uso de este tipo de organización de estructuras de nodos HTML, en el caso de necesitar un diseño más complejo o un comportamiento diferente, resulta muy incómodo.

Los estilos que se aplican desde fuera del Shadow DOM tampoco afectan a su interior, por lo que la única manera de cambiar el estilo interno es utilizando las variables de css que proporciona Ionic o utilizando el selector "::part" para adentrarse en los componentes nativos que forman este nodo. Esto hace que sea más viable crear el elemento customizado desde cero utilizando nodos simples de HTML.

El caso que más ha evidenciado este comportamiento ha sido la maquetación del selector de números, un tipo de pregunta de los autorregistros.

Por último, mencionar que, para evitar complicar los archivos css, se han utilizado las clases nativas de los CSS Utilities de Ionic, las cuales cubren una gran variedad de casos de estilos CSS, como transformadores de texto, posiciones, márgenes, etc…

## 9.3 Flujo de la aplicación

La primera versión de la aplicación incluye todas las pantallas mencionadas en el apartado anterior, sin embargo, el uso de datos dinámicos obtenidos por peticiones HTTP en combinación con los efectos de las transiciones por defecto de Ionic daban una mala sensación a la hora de navegar por la aplicación.

Muchos datos aparecían “de repente” tras estar la página en blanco, habían elementos que saltaban porque otros tardaban en aparecer y ,por lo general, el efecto de transición de fundido no era del agrado de la empresa.

Para solucionar este problema se ha instalado el plugin Native Page Transitions, el cual utiliza aceleración por hardware de forma nativa para transicionar de una forma más elegante. Además, este plugin permite personalizar muchos parámetros, como la dirección, el efecto, delay, mantener pixeles fijos, etc...

También se detectó que el controlador de navegación de Ionic, NavController, aplicaba una animación a la vez que se traicionaba con el plugin instalado, por lo que se han desactivado todas las animaciones de este controlador para mejorar el rendimiento y mantener el efecto limpio de Native Page Transitions.

Para el uso del plugin mencionado es necesario declarar un objeto de opciones para aplicar durante la transición, pero, con el fin de no declarar este objeto demasiadas veces y mantener una homogeneidad entre los mismos, se ha completado el servicio de transiciones, TransitionService, con las definiciones base.

Una vez resuelto el problema del efecto de la transición, se ha incluido en las pantallas más problemáticas en cuanto a elementos dinámicos una estructura skeleton. Este tipo de estructuras son muy utilizadas en la actualidad y funcionan como un placeholder de los datos hasta que éstos llegan y pueden ser utilizados y renderizados. Ionic trae un tipo de nodo llamado ion-skeleton-text que sirve para este propósito.

La siguiente figura muestra cómo se verían las páginas de listado y completado de autorregistros con skeleton antes de recibir los datos dinámicos de la API. Cabe destacar que tienen una animación y el resultado es bastante bueno.

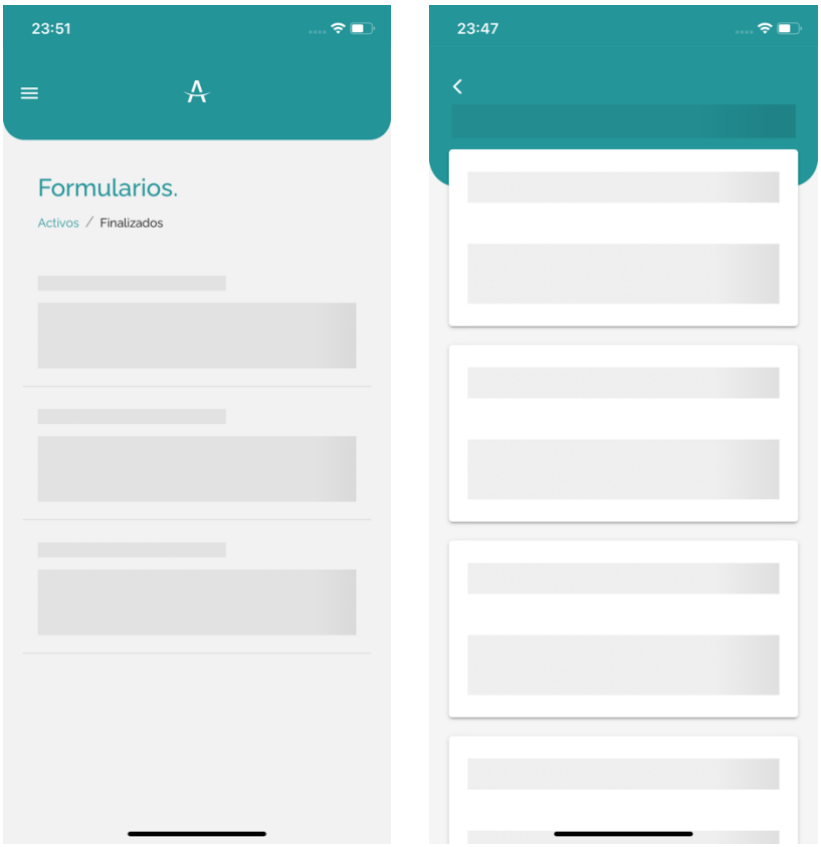


*Figura 40. Pantalla de Listado y Autorregistro con Skeleton*
*(Fuente propia)*

Para la navegación en la aplicación, se ha implementado un menú, el cual permite acceder a la mayoría de páginas desde el listado de autorregistros.

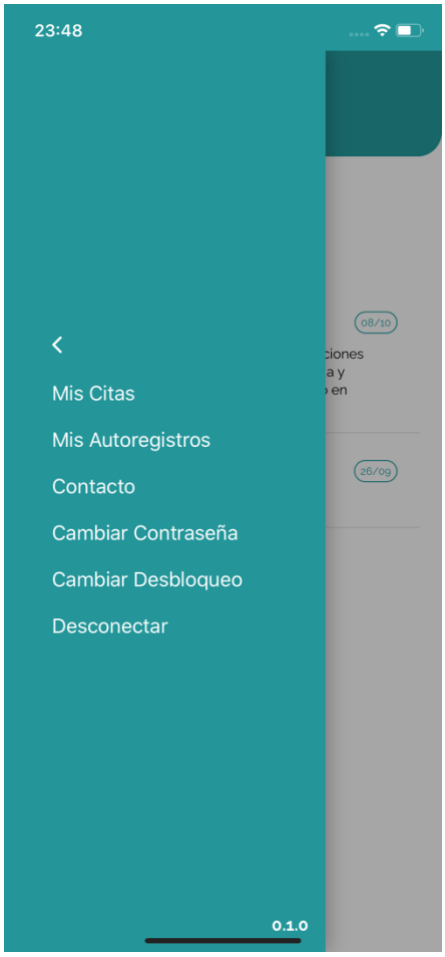


*Figura 41. Menu Lateral*
*(Fuente propia)*

Además, se ha implementado una serie de modales para darle feedback al usuario sobre el resultado de diferentes operaciones. Se tiene un modal de éxito, un modal de error y un modal de carga.

Los modales de éxito y error son componentes propios. En cambio, el modal de carga es un elemento de Ionic. Para su uso, los de éxito y error basta con importarlos a la página necesaria, pero el modal de carga es necesario definirlo en cada pantalla que se necesite. Para evitar este último problema, se ha completado un servicio, el LoadingService, el cual define el modal de carga solo una vez y este puede ser importado y utilizado de una forma similar a los de éxito y error.

A continuación, se adjunta una figura que representa estos modales:

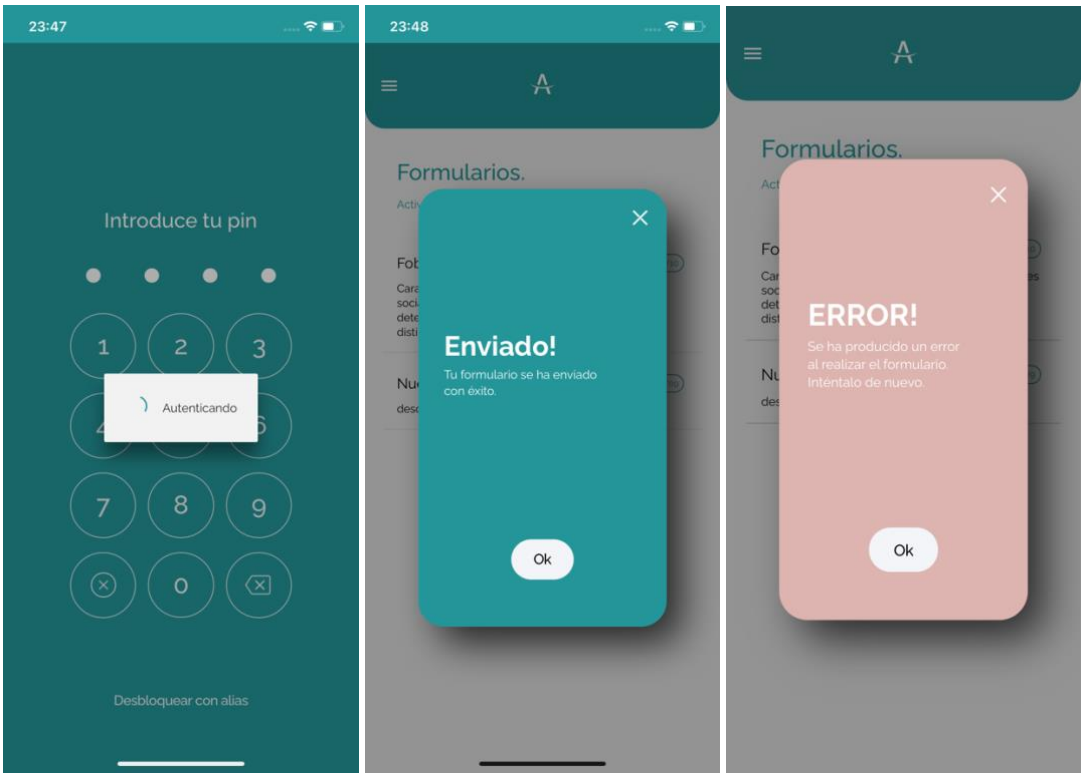


*Figura 42. Modales*
*(Fuente propia)*

## 9.4 Persistencia de Datos

Tal como se adelantó en el capítulo de Diseño, aunque solamente se guarden en el dispositivo unos datos mínimos del usuario y algunas variables de control de flujo, la persistencia resulta ser uno de los elementos más importantes para el correcto funcionamiento de la aplicación.

Los plugins utilizados para la implementación son la versión oficial de Ionic Storage y la versión actualizada y mantenida por la comunidad de Secure Storage.

Para ofrecer la mayor seguridad posible, se le da prioridad al uso de Secure Storage a pesar de sus limitaciones. Siempre que se cumplan las condiciones definidas en el capítulo de Diseño, el

usuario tendrá sus datos cifrados con el estándar AES y gozará de una seguridad robusta. Recordamos que estas condiciones son:

- En Android, tener un bloqueo de pantalla seguro, como pueden ser el PIN, una contraseña o el uso de alguna identificación biométrica.
- En iOS funcionará de manera correcta siempre que el usuario tenga activada una cuenta de apple para el dispositivo.

Por otra parte, el uso de Ionic Storage estará limitado a usuarios que explícitamente elijan una opción menos segura en el aviso que se mostrará al iniciar la aplicación.

Para gestionar la persistencia de manera transparente de cara a la aplicación, se importan ambos plugins dentro del StorageService y, dependiendo del tipo de persistencia que se utilice, se hacen las consultas y las escrituras sobre uno u otro. Esto es posible porque ambas formas de persistencias guardan los datos en formato clave-valor y la lógica queda bastante simplificada.

En cuanto al uso de la persistencia, se sigue el siguiente flujo:

- Al iniciar se comprueba la disponibilidad del Secure Storage:
    - Si está disponible y el KeyStore de Android no ha sido modificado, se recuperará esta base de datos.
    - Si está disponible pero el KeyStore de Android ha sido modificado, se mostrará un aviso informativo y se redirigirá al usuario al login, ya que pierde acceso a la base de datos previa y es necesario generar una nueva.
    - Si no está disponible, mostrará un aviso informativo aconsejando al usuario que active alguna forma segura de bloqueo de pantalla. En caso de aceptar, se abrirá la aplicación de ajustes del sistema. También tendrá las opciones de salir de la aplicación o seguir usando la aplicación con una base de datos sin cifrado.

- Si el usuario escoge la opción de base de datos sin cifrado, se dejará de comprobar la disponibilidad del Secure Storage hasta que se desconecte de la aplicación y vuelva a hacer login.
- Si el usuario utiliza Secure Storage y desactiva el bloqueo de pantalla, se le volverá a mostrar el aviso informativo con las opciones previamente mencionadas.

Cabe destacar que, independientemente del plugin utilizado, las operaciones con las bases de datos son rápidas y no ha habido ningún problema de estabilidad durante el testeo realizado. Es cierto que la opción del plugin de Offline Storage es la mejor opción para una aplicación de este tipo, pero una buena gestión de Secure Storage cubre de manera satisfactoria la mayoría de casos en los que se necesita una persistencia segura.

## 9.5 Bloqueo de la Aplicación y Seguridad

Para no depender exclusivamente del bloqueo de pantalla del dispositivo, se ha desarrollado un sistema de bloqueo de la aplicación.

Este sistema se encarga de bloquear la aplicación al salir de la misma, ya sea finalizando su actividad o enviándola a segundo plano. Además, es obligatorio y no puede ser desactivado.

Las tres formas de desbloqueo son:

- Un PIN de cuatro dígitos.
- Identificación biométrica: huella o Face ID.
- Alias y contraseña.

A continuación, se procede a explicar en más detalles el funcionamiento y la implementación de las mismas.

### 9.5.1 PIN

Como ya se adelantó en la sección correspondiente a la seguridad en el capítulo de Diseño, ninguno de los componentes de PIN existentes se adecuaba a las especificaciones de la aplicación, por lo que se ha desarrollado un componente propio.

La parte gráfica se ha implementado siguiendo la línea de diseño de las interfaces facilitadas por la empresa. Los elementos visuales utilizados han sido botones para el teclado virtual y círculos customizados por CSS para el indicador de la parte superior. En la *Figura 43. Desbloqueo por PIN* se puede ver el resultado.

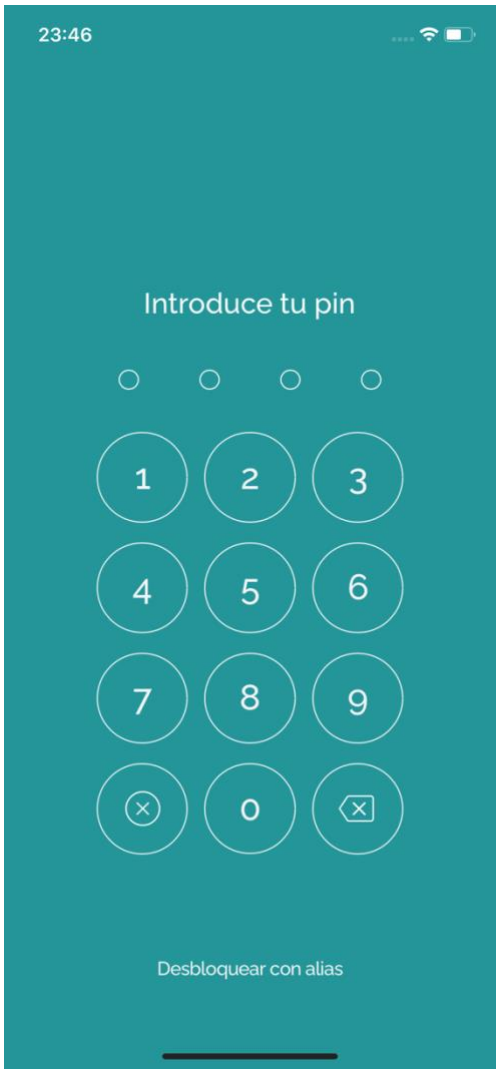


*Figura 43. Desbloqueo por PIN*
*(Fuente propia)*

Para su reutilización en distintos contextos, este componente incluye variables opcionales de entrada y emisión de eventos. Las posibles inputs son:

- Booleano: para el indicador de la parte superior, utilizar los círculos o mostrar los números del PIN.
- Cadena de texto: el título de la parte superior.
- Entero de 4 dígitos: el número a comprobar. Si es -1, no se comprobará el número al introducir 4 dígitos con el teclado virtual.

  En caso contrario, se comprobará y se emitirá un evento con un booleano, indicando si coincide o no. En el caso de no coincidir, se animará la parte de círculos superiores para darle feedback al usuario.

Esta forma de desbloqueo resulta bastante útil y rápida y resulta ser la más indicada en el caso de no tener disponible la opción de huella o Face ID.

### 9.5.2 Identificación biométrica

Para dotar a la aplicación de esta funcionalidad y completar el sistema de bloqueo de la aplicación, se usa el plugin Fingerprint AIO [7].

Fingerprint AIO es bastante simple. Su primera versión incluía solamente el soporte de autenticación por huella, ya sea en Android con sus numerosas variantes o en iOS con Touch ID, pero la última versión incluye el soporte de Face ID para los nuevos dispositivos de Apple.

Cabe destacar que no incluye soporte para el reconocimiento facial de Android. En la empresa se ha valorado esta carencia pero se le ha dado el visto bueno al plugin. El reconocimiento facial en Android depende mucho del dispositivo y, en muchos casos, las cámaras que se encargan de este proceso son de baja calidad, pudiendo desbloquear dispositivos con fotografías.

Para utilizar este plugin es necesario comprobar antes la disponibilidad de la autenticación biométrica, ya que solamente se podrá utilizar si ésta está activa.

Una vez realizada la comprobación, si se puede seguir, se muestra el controlador nativo de autenticación, ya sea huella, Touch ID o Face ID, lo que permite únicamente autenticar al usuario.

La siguiente figura muestra los controladores nativos en los casos de huella en Android y Face ID en iOS:

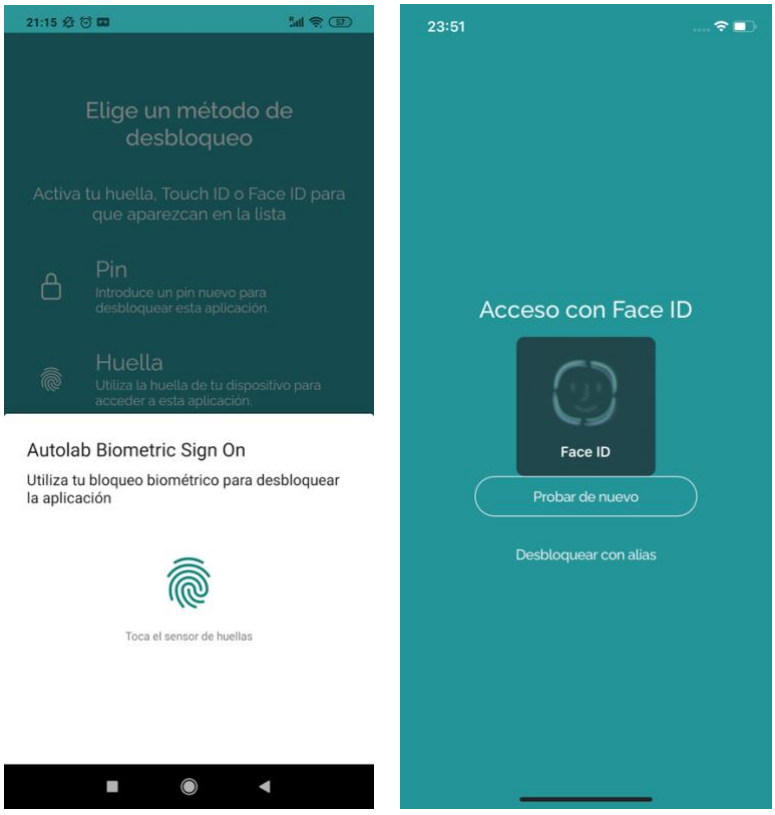


*Figura 44. Desbloqueo por Huella y FaceID*
*(Fuente propia)*

Esta forma de bloqueo es la más rápida y es la que se recomienda si el bloqueo de pantalla por autenticación biométrica está activado.

### 9.5.3 Alias y contraseña

Esta se trata de la opción más costosa en cuanto a tiempo. Se trata de la misma forma de desbloqueo que el login inicial de la aplicación, utilizando el alias y la contraseña del usuario para acceder.

Además de ser una opción más para el bloqueo de la aplicación, también es la opción por defecto en el caso de que alguna de las opciones anteriores falle o sea imposible de realizar. Durante el desbloqueo con las demás opciones, se mostrará un botón que permitirá redirigir a esta opción.

Para seguir la misma línea de diseño de las demás formas de bloqueo, se ha creado un nuevo componente. De esta manera, la otra pantalla de login será utilizada solamente para el acceso inicial a la aplicación.

En la *Figura 45. Desbloqueo por Alias y Contraseña* podemos observar el diseño de este nuevo componente:

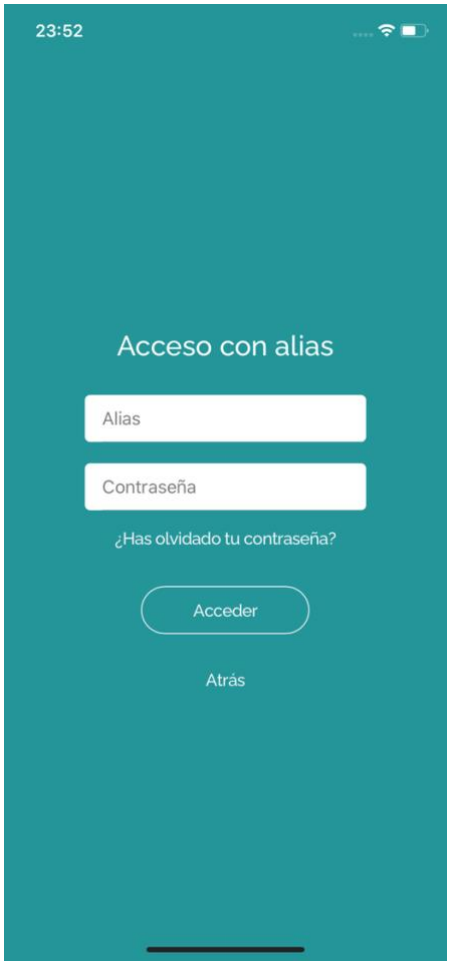


*Figura 45. Desbloqueo por Alias y Contraseña*
*(Fuente propia)*

### 9.5.4 Flujo general

Para poder gestionar y escoger el sistema de bloqueo preferido, se ha creado una página nueva. Se muestra un listado con las opciones disponibles y contiene un slider con el recorrido necesario para completar el proceso.

Las opciones de PIN y alias siempre estarán disponibles, pero la opción biométrica tendrá que estar activada para el bloqueo del dispositivo. La *Figura 46. Elegir Bloqueo de Aplicación* muestra el primer paso de este proceso.

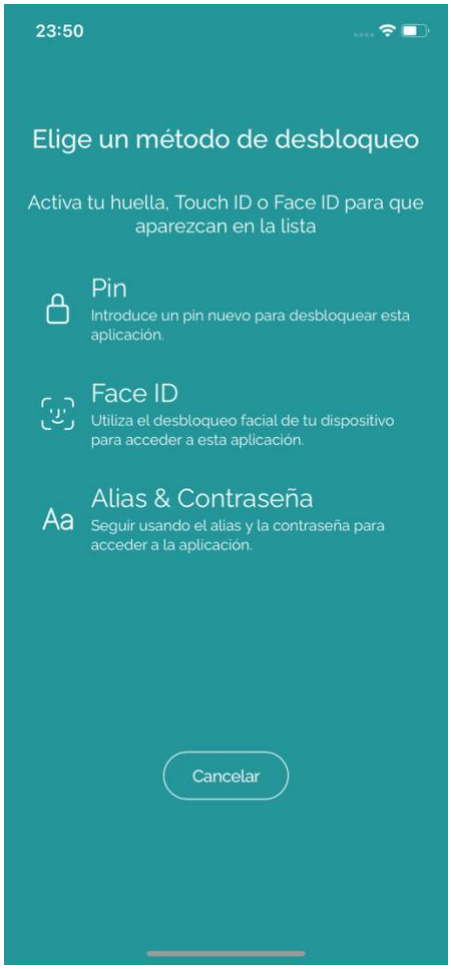


*Figura 46. Elegir Bloqueo de Aplicación*
*(Fuente propia)*

Por lo general, el flujo de este sistema es el siguiente:

- Al iniciar la aplicación y hacer login, se obliga al usuario a elegir entre una de las formas de bloqueo disponibles.
- Cuando la configuración se haya completado de manera correcta, se llevará al usuario al listado de autorregistros.
- Si el usuario sale de la aplicación, ya sea cerrándola o enviándola a segundo plano, ésta se bloqueará con la forma de bloqueo escogida.
- Si el usuario no recuerda su PIN o existen problemas durante la autenticación biométrica, podrá acceder introduciendo su alias y contraseña.
- Dentro de la aplicación, si el usuario quiere cambiar su forma de bloqueo, se dirigirá a la página correspondiente accediendo a ella a través del menú. Se pedirá al usuario que desbloquee la aplicación con la forma de bloqueo actual como confirmación y podrá seguir el proceso para escoger una nueva forma de bloqueo.
- Si el usuario activa inicialmente la identificación biométrica y la desactiva desde los ajustes de la aplicación, como esta forma ya no está disponible para el bloqueo de la aplicación, se utiliza la opción por defecto, el acceso con alias y contraseña.

## 9.6 Firebase y Notificaciones Push

Para Ionic y Apache Cordova hay varios plugins para activar las funcionalidades de Firebase, pero se repite la misma situación que con otros muchos plugins: las versiones oficiales han dejado de ser mantenidas por Ionic y han pasado a manos de la comunidad. En muchas ocasiones, esto causa que la implementación no sea la más óptima y que aparezcan incompatibilidades con otros plugins.

El plugin FirebaseX representa perfectamente esta situación. Está basado en el plugin oficial de Ionic y se trata del más completo, dotando a la aplicación de todas las librerías importantes de Firebase como analíticas, Cloud Messaging, autenticación y persistencia en la nube.

A pesar de todas sus bondades, al ser instalado en la aplicación se ha detectado los siguientes problemas:

- Necesita configuraciones para todas las librerías que trae, llenando la consola de avisos en caso de no tenerlas.
- Tiene un impacto negativo en el rendimiento de la aplicación.
- Obliga a subir la versión mínima de Android a la API 21 por la forma que tiene de importar todas las librerías con Gradle.
- La implementación de notificaciones con Cloud Messaging no es compatible con la implementación de Ionic, provocando que éstas no se puedan abrir.

Todos estos motivos han llevado a descartar este plugin y utilizar otro, cordova-plugin-fcm-with-dependecy-updated, también mantenido por la comunidad. Este plugin trae solamente la funcionalidad de Cloud Messaging, siendo necesario complementar el uso de analíticas con otro plugin.

Cordova-plugin-fcm-with-dependecy-updated funciona de la manera esperada y la implementación de las notificaciones es correcta. Además, contiene funciones bastante útiles para hacer comprobaciones y requerir permisos de mensajería push, obligatorio en iOS.

La implementación en Ionic es bastante sencilla:

- Se comprueban los permisos y se requieren de ser necesarios
- Si hay permisos, se obtiene un token único por instalación.
- Se envía el token al servidor para que tenga la referencia de los dispositivos a los cuales se enviarán las notificaciones push. El servidor los almacena, por lo que el usuario podrá recibir las notificaciones desde varios dispositivos.
- Se inicia un listener para recibir notificaciones.
- Al recibirlas, dependiendo del tipo de notificación que sea, se realizará una operación distinta.
- Los tipos de notificaciones que pueden recibir un paciente son:
    - Recibir nuevos autorregistros: la notificación contiene los datos de dichos autorregistros.
    - Confirmación de cita (por el psicólogo): la notificación contiene el id de la cita a activar en la aplicación.
    - Recordatorio de cita un día antes de la fecha en cuestión.

En cuanto a la consola de Firebase, se ha creado un proyecto nuevo con dos aplicaciones, una para Android y otra para iOS. Durante el proceso, se han descargado los 2 archivos de configuración de Firebase, uno para cada plataforma y éstos han sido movidos a la carpeta raíz del proyecto.

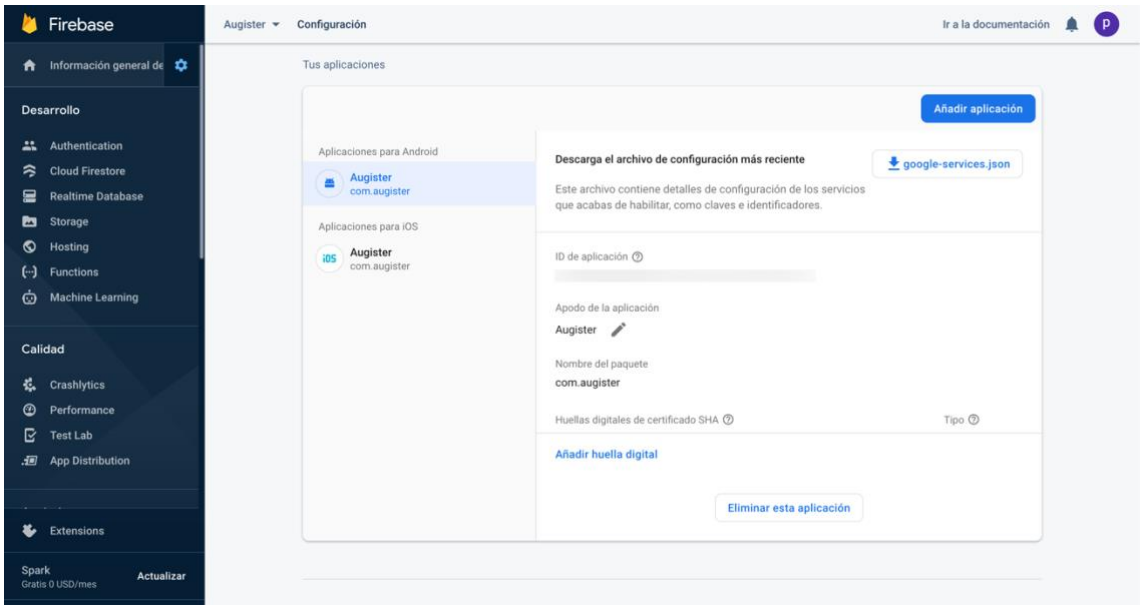


*Figura 47. Apps de Augister en Firebase*
*(Fuente propia)*

Como el plugin de cordova-plugin-fcm-with-dependecy-updated espera éstos archivos para generar las plataformas de Ionic de forma correcta, ha sido necesario borrar y volver a generarlas para que el plugin pudiera recoger los archivos y colocarlos en los directorios necesarios.

Además, para el funcionamiento del Cloud Messaging en iOS, ha sido necesario aportar una clave APN, la cual ha sido generada desde el panel de desarrollador de Apple de la cuenta de la empresa.

Una vez realizadas todas las configuraciones, se ha probado el servicio realizando una llamada con un cliente de peticiones HTTP REST, Postman [14]. Como se puede observar en las siguientes, su funcionamiento es correcto.

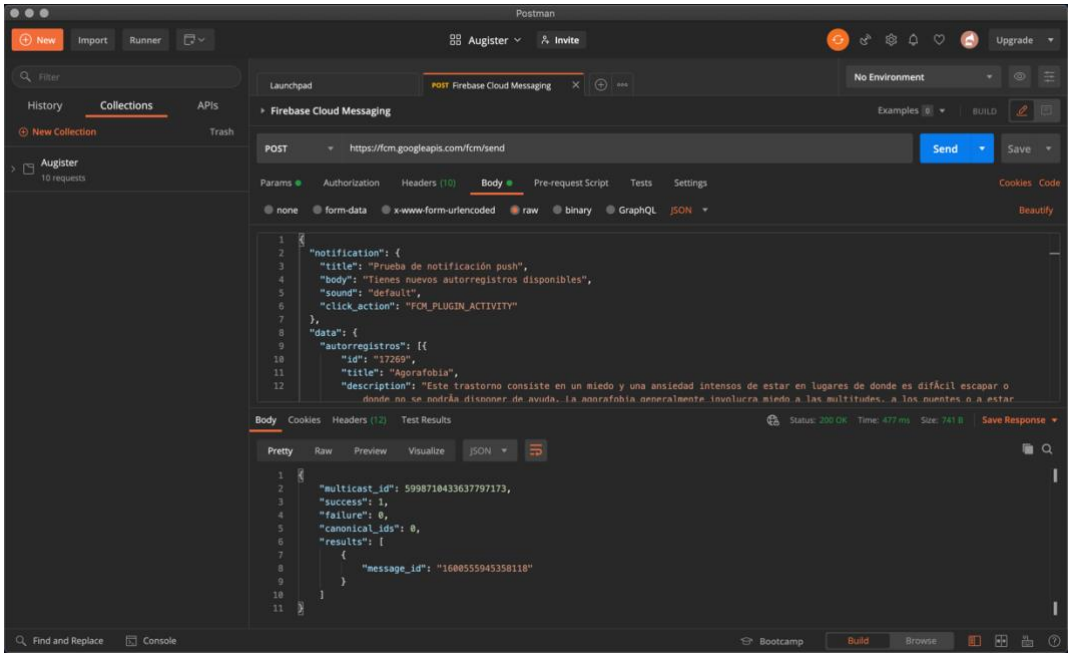


*Figura 48. Enviar Notificación Push por Postman*
*(Fuente propia)*


*Figura 49. Ejemplo de Notificación Push*
*(Fuente propia)*

## 9.7 Integración con la API de Augister

La integración con la API de Augister ha sido relativamente sencilla. Los endpoints y la especificación de los mismos han sido proporcionados por la empresa.

Para centralizar todas las peticiones y no importar el cliente http de Angular en todas las páginas, se ha completado el servicio de datos, el DataService.

Todas las llamadas han sido separadas por funciones y se han realizado las configuraciones necesarias para su correcto funcionamiento:

- Es necesario adjuntar un objeto de configuración de cabeceras a todas las peticiones para evitar que las llamadas sean guardadas en caché. Ionic tiene una política de guardado de llamadas http en caché, por lo que es necesario deshacernos de este comportamiento para el caso de esta aplicación.
- Se define un timeout de 20 segundos para todas las llamadas.
- Todas las llamadas son transformadas a promesas para manejar los resultados de una forma más sencilla.

El único problema encontrado durante la integración a la API ha sido CORS, el Intercambio de Recursos de Origen Cruzado de HTTP. Como el dominio de la aplicación durante las pruebas es localhost, ha sido necesario configurar el servidor de Augister para que aceptase peticiones de este tipo.

## 9.8 Publicación en los Marketplaces

Una vez llegados a este punto, se tiene una versión estable de la aplicación con las principales funcionalidades implementadas y listas para verificar. El equipo de la empresa probará la aplicación y se realizarán los procesos de verificación necesarios para garantizar el correcto funcionamiento de la misma.

A falta de este proceso de pruebas, se sube la aplicación a los dos Marketplaces (Google Play y App Store) en un estado alfa, permitiendo la sencilla distribución entre los integrantes del equipo de la empresa.

Para ello, se ha compilado la aplicación para Android e iOS desde Ionic utilizando las etiquetas de producción y release. Esta compilación es mucho más exhaustiva y ha permitido encontrar pequeños errores que, si bien no alteran el funcionamiento de la aplicación, afectan a su rendimiento y estabilidad.

Para el proyecto compilado de Android se ha creado un KeyStore de forma manual, se ha firmado la APK de release con la misma y se ha comprimido con la herramienta zipalign. El paquete de instalación resultante ha sido subido al canal Alpha de la ficha de la aplicación en Google Play Console y se ha distribuido al equipo.


*Figura 50. Ficha de Augister en Google Play*
*(Fuente propia)*

En cuanto al proyecto de iOS, se ha realizado el procedimiento estándar de distribución mediante Xcode y TestFlight [10]. Con Xcode se ha subido la aplicación a App Store Connect y una vez procesado el paquete, se ha distribuido a la lista de pruebas del equipo mediante TestFlight.

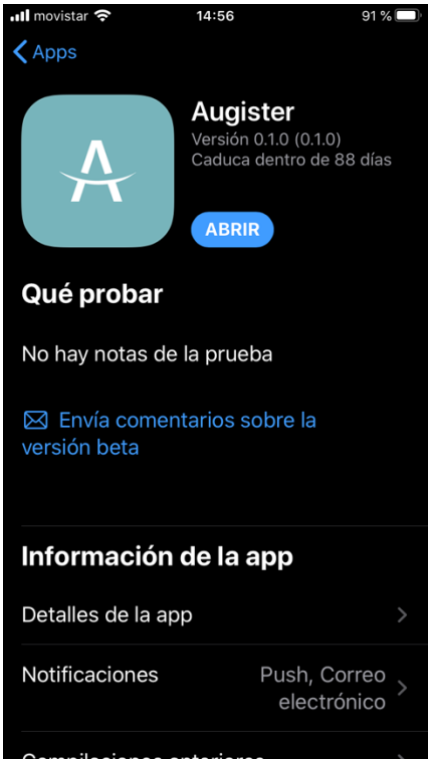


*Figura 51. Augister en TestFlight*
*(Fuente propia)*

En su estado actual, la aplicación se encuentra en un estado Alpha, a espera de que las validaciones de la misma concluyan y pueda ser probada por usuarios reales para recibir feedback.

# 10. Conclusiones

Este apartado recoge las conclusiones extraídas a lo largo del desarrollo de este Trabajo Fin de Máster.

## 10.1 Estado de la aplicación

Analizando la situación actual de la aplicación desarrollada, se puede afirmar que se ha cumplido satisfactoriamente con los objetivos principales planteados en el capítulo de Objetivos.

La aplicación cuenta con el perfil completo del paciente, conteniendo la lógica necesaria para permitir el acceso a los autorregistros que le han sido asignados, completarlos y enviar las respuestas al panel del psicólogo, pedir una cita y consultar los datos de contacto del consultorio o psicólogo al que acude.

Las pantallas implementadas tienen unas interfaces sencillas y un diseño moderno, adaptándose de manera correcta y coherente al servicio web de Augister.

Asimismo, se han tenido en cuenta las tecnologías actuales para almacenar los datos sensibles del paciente de manera segura y reforzar la seguridad con un sistema de bloqueo a la aplicación.

A pesar de los buenos resultados y haber cumplido con los objetivos principales esperados, la aplicación no está completa, ya que faltan principalmente el perfil de psicólogo y la gestión de autorregistros. En el capítulo de Trabajo Futuro se profundiza en este aspecto.

## 10.2 Metodología

En cuanto a la metodología utilizada, ésta ha dado resultados y ha habilitando un canal de comunicación y feedback constante. Gracias a esto se ha podido alcanzar satisfactoriamente los objetivos principales y se ha construido un producto que cumple con su finalidad y se adapta al servicio de Augister.

Dada la crisis sanitaria actual y la consecuente bajada de rendimiento de la mayoría de empresas, no se ha podido cumplir en totalidad con las fechas previstas en el capítulo de Planificación. Además, como ya se ha comentado previamente, no se han alcanzado algunos de los objetivos secundarios planteados durante la primera fase del proyecto.

A pesar de estos inconvenientes, la combinación de metodologías utilizadas se han adaptado perfectamente a la situación. Aunque el ritmo de trabajo no haya sido del todo constante, se ha tenido en todo el momento una buena comunicación con la empresa y el feedback ha estado presente en cada iteración que se ha llevado a cabo.

Cabe destacar que esta metodología, aunque haya sido especialmente útil en esta situación, no se puede adaptar en su totalidad a próximos proyectos, ya que se ha pensado teniendo en cuenta unas circunstancias muy específicas. No obstante, se recoge una valiosa experiencia a tener en cuenta para la planificación y desarrollo de nuevos proyectos en el futuro.

## 10.3 Ionic

Ionic es la principal tecnología que se ha utilizado para el desarrollo de la aplicación multiplataforma de este proyecto. Se trata de uno de los requisitos de la empresa y es también una de las tecnologías que se han visto a lo largo del Máster Universitario en Desarrollo de Software para Dispositivos Móviles.

Este framework se ha adaptado bastante bien a los requerimientos y especificaciones del proyecto. Además, la combinación con Angular como framework web ha resultado bastante útil, especialmente por la familiaridad del alumno con el mismo.

Aunque el proceso del desarrollo haya sido por lo general bastante satisfactorio, se han detectado una serie de puntos débiles, los cuales, en ocasiones, han dificultado la implementación.

Ionic tiene a su disposición una gran variedad de plugins, la mayoría de calidad, cumpliendo con los requisitos de un desarrollo robusto. No obstante, muchos plugins que previamente eran mantenidos de manera oficial, han pasado a ser mantenidos y actualizados por la comunidad. Aunque esto no sea del todo negativo, sí que influye en la calidad de los mismos, teniendo implementaciones incompletas o no actualizadas a las últimas versiones del framework.

En cuanto a la documentación, Ionic mantiene activas varias versiones de la misma. El problema aparece cuando las últimas versiones no están completas, no contienen todos los detalles de los atributos de ciertos componentes o no explican el funcionamiento completo de los mismos, siendo necesario recurrir a versiones anteriores o informarse directamente del repositorio Git.

Si a esto se le suma la problemática del diseño que se explicó en detalle en el capítulo de Implementación y Resultados, me despierta la curiosidad por probar otros frameworks actuales de desarrollo de aplicaciones multiplataforma, como pueden ser Flutter o React Native, y comprobar si ofrecen una mejor experiencia de programación y diseño de interfaces.

# 11. Trabajo futuro

Como ya se ha comentado previamente, la aplicación no se encuentra completamente finalizada. A continuación, se definen los pasos a seguir para completar su desarrollo fuera del alcance de este Trabajo Fin de Máster.

En su estado actual, la aplicación tiene un perfil de paciente funcional, pero es necesario completarlo añadiendo el sistema de citas. Esta funcionalidad se encuentra en reconstrucción en la parte del servidor y lo más probable es que cambie el flujo de pedir cita y obtener la confirmación por parte del psicólogo.

Cuando esta funcionalidad esté lista en el servicio web, se adaptará la lógica actual de citas de la aplicación al nuevo flujo, actualizando el DataService con los nuevos endpoints del servidor.

Asimismo, cuando la funcionalidad de notificaciones push esté completamente desarrollada en la parte del servidor, se adaptará el sistema actual de la aplicación a los mensajes a recibir. Cabe destacar que la aplicación, en su estado actual, sí que recibe mensajes y notificaciones push y la lógica ha sido preparada para recibir los mensajes del servidor.

Como la aplicación se encuentra publicada en los marketplaces en un estado Alpha, para completar su paso a producción será necesario realizar las pruebas de usabilidad, recibiendo feedback de varios usuarios, los cuales probarán la aplicación de forma anticipada. En este paso se llevará a cabo el plan de pruebas definido en el apartado de Pruebas y Validación del capítulo de Diseño.

Una vez completados todos estos pasos, podrá ser publicada como Release en los Marketplaces y estará disponible para los pacientes de los psicólogos registrados en Augister.

El último gran paso sería la implementación del perfil de psicólogo, permitiéndole el acceso y gestión de sus autorregistros, asignarlos a sus pacientes, gestionar sus pacientes y gestionar las citas. De nuevo, se repetirán las pruebas y las validaciones para su publicación en los marketplaces.

# Bibliografía


1. *Ionic.* [Online] https://ionicframework.com.
2. *Angular.* [Online] https://angular.io.
3. *Flutter.* [Online] https://flutter.dev.
4. *React Native.* [Online] https://reactnative.dev.
5. *Ionic Offline Storage.* [Online] https://ionicframework.com/enterprise/offline-storage.
6. *Secure Storage para Ionic.* [Online] https://github.com/mibrito707/cordova-plugin-secure-storage-echo.
7. *Fingerprint AOI para Ionic.* [Online] https://github.com/NiklasMerz/cordova-plugin-fingerprint-aio.
8. *A Study of Encryption Algorithms AES, DES and RSA for Security.* **Dr. Prerna Mahajan, Abhishek Sachdeva. 2013.** 15-E, Massachusetts : Global Journal of Computer Science and Technology, 2013, Vol. 13. ISSN: 0975-4172.
9. App para psicólogos. *iGrade.* [Online] https://apps.apple.com/es/app/igrade-para-psicólogo/id572563154.
10. Beta Testing Made Simple with TestFlight. *Apple Developer.* [Online] https://developer.apple.com/testflight/.
11. **Drumond, Claire.** ¿Qué es SCRUM? *Atlassian.* [Online] https://www.atlassian.com/es/agile/scrum.
12. Especificación de Requisitos según el estándar de IEEE 830. [En línea] https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf.
13. Firebase Cloud Messaging. *Firebase.* [Online] https://firebase.google.com/docs/cloud-messaging.
14. Herramienta para análisis de API REST. En forma de plugin para Chrome y aplicación. *Postman.* [En línea] https://www.getpostman.com/.
15. **Julio Alonso-Arévalo, José Antonio Mirón-Canelo. 2017.** Aplicaciones móviles en salud: potencial, normativa de seguridad y regulación. *Scielo.* [Online] Julio 2017.

http://scielo.sld.cu/scielo.php?pid=S2307-21132017000300005&script=sci_arttext&tlng=pt.

16. Libro de Estilos para la redacción de trabajos TFG/TFM de la EPS. *Escuela Politécnica Superior (Universidad de Alicante).* [En línea] https://maktub.eps.ua.es/servicios/gestorContenidos/contenidos/normativaEPS/Pdf/9910.pdf.

17. Lienzo Lean Canvas explicado. *Innokabi.* [Online] https://innokabi.com/lienzo-lean-canvas-el-lienzo-de-los-emprendedores/.

18. **2012.** OpenUp. *EcuRed.* [Online] Septiembre 24, 2012. https://www.ecured.cu/OpenUp.

19. **Pixel78.** Un Estudio Creativo situado en Benidorm. *Pixel78.* [Online] https://www.pixel78.com.

20. **—.** Un servicio de autorregistros online. *Augister.* [Online] https://www.augister.com.

21. Plataforma para la psicología y psiquiatría online. *Mentavio.* [Online] https://www.mentavio.es.

22. Plataforma para psicólogos para la gestión de pacientes. *Psicoreg.* [Online] https://psicoreg.com.

23. **Provincias, Las. 2017.** La Psicologia, una necesidad en la sociedad actual. *Col·legi Oficial de Psicologia de la Comunitat Valenciana.* [Online] Febrero 21, 2017. https://www.cop-cv.org/noticia/10817-la-psicologia-una-necesidad-en-la-sociedad-actual.

24. *Research and implementation of RSA algorithm for encryption and decryption.* **Xin Zhou, Xiaofei Tang. 2011.** Harbin, Heilongjiang, Harbin : IEEE, 2011. ISBN: 978-1-4577-0399-7.

25. Software de administración de proyectos . *Trello.* [Online] https://trello.com/.

26. Software for Administering Psychological Questionnaires to Patients Remotely. *NovoPsych.* [Online] https://novopsych.com.au.

27. **Zurkus, Kacy. 2019.** Who is managing the security of medical management apps? *MalwareBytes Labs.* [Online] Abril 14, 2019. https://blog.malwarebytes.com/101/2019/04/managing-security-medical-management-apps/.